中山大學
SUN YAT-SEN UNIVERSITY

2012级《多元统计分析与数据挖掘》

第2周

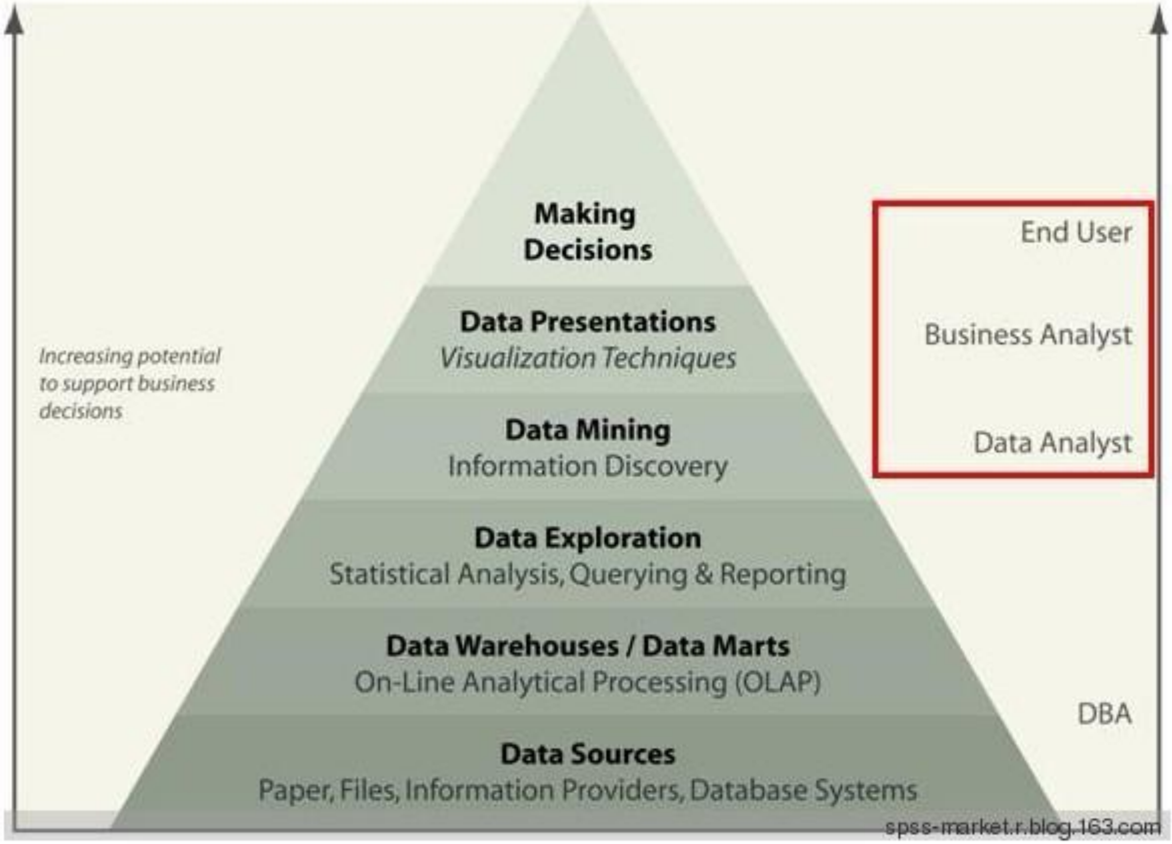
2015.3.10

中山大学数学与计算科学学院 黄志洪

课程介绍

- 上课时间，地点：周二，周四上午1-3节，1308。上课12周，至5月底结束
- 课程需要：计算机，上网环境，英语阅读能力
- 电子作业：书面，互动，大作业。作业，比赛、互动优异者可以加分
- 课程资源和作业平台（加入口令为“sysu@2015”）：http://www.dataguru.cn/myclass.php?mod=new_basicforlesson&op=basic&lessonid=345
- 考试：3-3.5-3.5，期中考与期末考
- 老师的联系方式：手机13802502960，邮件stswzh@sysu.edu.cn，QQ1829118
- 课程交流qq群：414907025。加入时请注明自己的学号，专业，姓名以便审核

多层模型


Making
Decisions
Data Presentations
Visualization Techniques
Data Mining
Information Discovery
Data Exploration
Statistical Analysis, Querying & Reporting
Data Warehouses / Data Marts
On-Line Analytical Processing (OLAP)
Data Sources
Paper, Files, Information Providers, Database Systems
Increasing potential
to support business
decisions
End User
Business Analyst
Data Analyst
DBA
spss-market.r.blog.163.com

软件

- R
- Weka
- Matlab
- Python
- 参考：http://blog.csdn.net/hzxhan/article/details/8548801

R

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。S-PLUS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，两个软件有一定的兼容性。

2015.3.10

R

- R is free

R是用于统计分析、绘图的语言和操作环境。R是属于GNU系统的一个自由、免费、源代码开放的软件，它是一个用于统计计算和统计制图的优秀工具。

R是一套完整的数据处理、计算和制图软件系统。其功能包括：数据存储和处理系统；数组运算工具（其向量、矩阵运算方面功能尤其强大）；完整连贯的统计分析工具；优秀的统计制图功能；简便而强大的编程语言：可操纵数据的输入和输入，可实现分支、循环，用户可自定义功能。

R是一个免费的自由软件，它有UNIX、LINUX、MacOS和WINDOWS版本，都是可以免费下载和使用的,在那儿可以下载到R的安装程序、各种外挂程序和文档。在R的安装程序中只包含了8个基础模块，其他外在模块可以通过CRAN获得。

R官方网站地址：http://www.r-project.org

R

R的特点

1．有效的数据处理和保存机制。

2．拥有一整套数组和矩阵的操作运算符。

3．一系列连贯而又完整的数据分析中间工具。

4．图形统计可以对数据直接进行分析和显示，可用于多种图形设备。

5．一种相当完善、简洁和高效的程序设计语言。它包括条件语句、循环语句、用户自定义的递归函数以及输入输出接口。

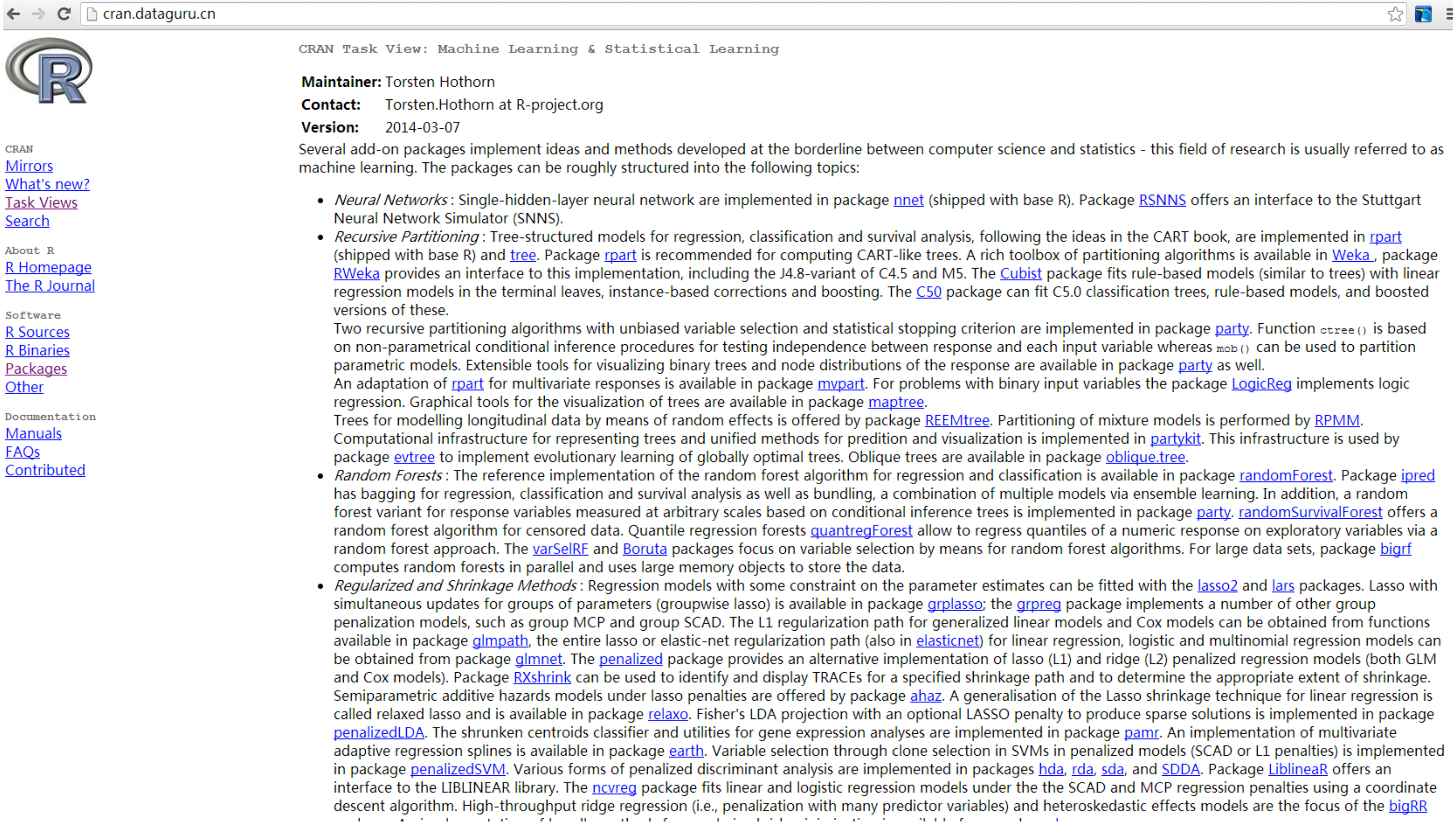
6．R语言是彻底面向对象的统计编程语言。

7．R语言和其它编程语言、数据库之间有很好的接口。

8．R语言是自由软件，可以放心大胆地使用，但其功能却不比任何其它同类软件差。

9．R语言具有丰富的网上资源

R的CRAN Task View



CRAN Task View: Machine Learning & Statistical Learning

Maintainer: Torsten Hothorn
Contact: Torsten.Hothorn at R-project.org
Version: 2014-03-07

Several add-on packages implement ideas and methods developed at the borderline between computer science and statistics - this field of research is usually referred to as machine learning. The packages can be roughly structured into the following topics:

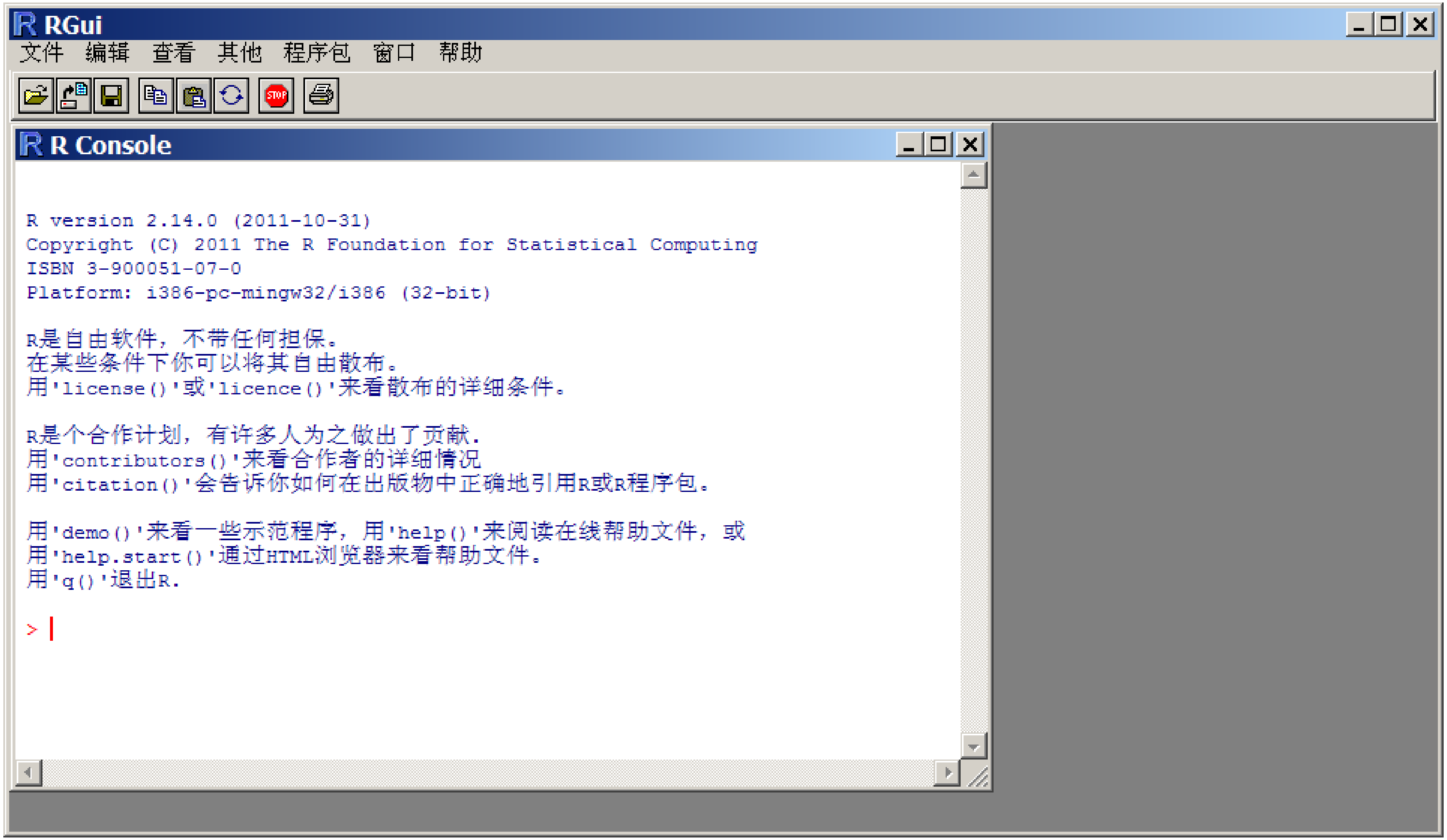
- *Neural Networks* : Single-hidden-layer neural network are implemented in package nnet (shipped with base R). Package RSNNS offers an interface to the Stuttgart Neural Network Simulator (SNNS).
- *Recursive Partitioning* : Tree-structured models for regression, classification and survival analysis, following the ideas in the CART book, are implemented in rpart (shipped with base R) and tree. Package rpart is recommended for computing CART-like trees. A rich toolbox of partitioning algorithms is available in Weka , package RWeka provides an interface to this implementation, including the J4.8-variant of C4.5 and M5. The Cubist package fits rule-based models (similar to trees) with linear regression models in the terminal leaves, instance-based corrections and boosting. The C50 package can fit C5.0 classification trees, rule-based models, and boosted versions of these.
  Two recursive partitioning algorithms with unbiased variable selection and statistical stopping criterion are implemented in package party. Function `ctree()` is based on non-parametrical conditional inference procedures for testing independence between response and each input variable whereas `mob()` can be used to partition parametric models. Extensible tools for visualizing binary trees and node distributions of the response are available in package party as well.
  An adaptation of rpart for multivariate responses is available in package mvpart. For problems with binary input variables the package LogicReg implements logic regression. Graphical tools for the visualization of trees are available in package maptree.
  Trees for modelling longitudinal data by means of random effects is offered by package REEMtree. Partitioning of mixture models is performed by RPMM.
  Computational infrastructure for representing trees and unified methods for predition and visualization is implemented in partykit. This infrastructure is used by package evtree to implement evolutionary learning of globally optimal trees. Oblique trees are available in package oblique.tree.
- *Random Forests* : The reference implementation of the random forest algorithm for regression and classification is available in package randomForest. Package ipred has bagging for regression, classification and survival analysis as well as bundling, a combination of multiple models via ensemble learning. In addition, a random forest variant for response variables measured at arbitrary scales based on conditional inference trees is implemented in package party. randomSurvivalForest offers a random forest algorithm for censored data. Quantile regression forests quantregForest allow to regress quantiles of a numeric response on exploratory variables via a random forest approach. The varSelRF and Boruta packages focus on variable selection by means for random forest algorithms. For large data sets, package bigrf computes random forests in parallel and uses large memory objects to store the data.
- *Regularized and Shrinkage Methods* : Regression models with some constraint on the parameter estimates can be fitted with the lasso2 and lars packages. Lasso with simultaneous updates for groups of parameters (groupwise lasso) is available in package grplasso; the grpreg package implements a number of other group penalization models, such as group MCP and group SCAD. The L1 regularization path for generalized linear models and Cox models can be obtained from functions available in package glmpath, the entire lasso or elastic-net regularization path (also in elasticnet) for linear regression, logistic and multinomial regression models can be obtained from package glmnet. The penalized package provides an alternative implementation of lasso (L1) and ridge (L2) penalized regression models (both GLM and Cox models). Package RXshrink can be used to identify and display TRACEs for a specified shrinkage path and to determine the appropriate extent of shrinkage. Semiparametric additive hazards models under lasso penalties are offered by package ahaz. A generalisation of the Lasso shrinkage technique for linear regression is called relaxed lasso and is available in package relaxo. Fisher's LDA projection with an optional LASSO penalty to produce sparse solutions is implemented in package penalizedLDA. The shrunken centroids classifier and utilities for gene expression analyses are implemented in package pamr. An implementation of multivariate adaptive regression splines is available in package earth. Variable selection through clone selection in SVMs in penalized models (SCAD or L1 penalties) is implemented in package penalizedSVM. Various forms of penalized discriminant analysis are implemented in packages hda, rda, sda, and SDDA. Package LiblineaR offers an interface to the LIBLINEAR library. The ncvreg package fits linear and logistic regression models under the the SCAD and MCP regression penalties using a coordinate descent algorithm. High-throughput ridge regression (i.e., penalization with many predictor variables) and heteroskedastic effects models are the focus of the bigRR

Python

- Guido van Rossumzai 1989年创立了Python
- I wrote python!
- Python语言的特点
- NumPy
- SciPy http://scipy.org/install.html
- Matplotlib http://matplotlib.org/

2015.3.10

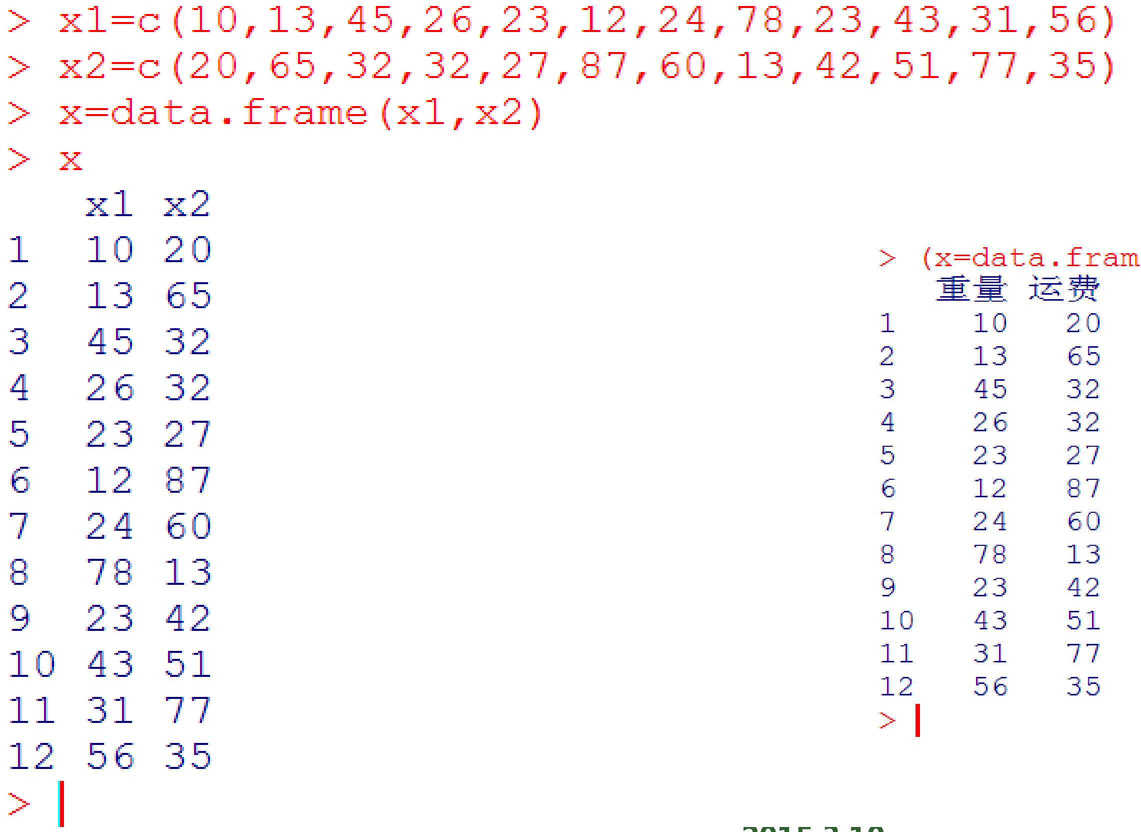
R语言



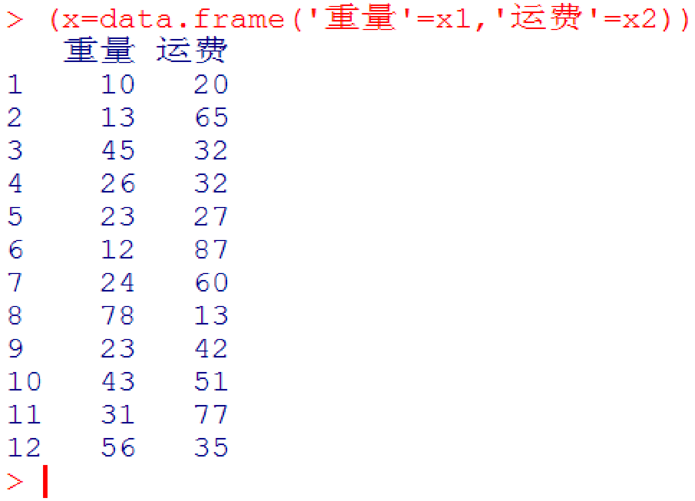
```
R version 2.14.0 (2011-10-31)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-bit)

R是自由软件，不带任何担保。
在某些条件下你可以将其自由散布。
用'license()'或'licence()'来看散布的详细条件。

R是个合作计划，有许多人为之做出了贡献.
用'contributors()'来看合作者的详细情况
用'citation()'会告诉你如何在出版物中正确地引用R或R程序包。

用'demo()'来看一些示范程序，用'help()'来阅读在线帮助文件，或
用'help.start()'通过HTML浏览器来看帮助文件。
用'q()'退出R.

>
```

数据的R语言表示——数据框

- 矩阵形式，但列可以不同数据类型
- 每列是一个变量，每行是一个观测值 ‘



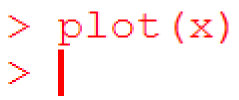
```
> x1=c(10,13,45,26,23,12,24,78,23,43,31,56)
> x2=c(20,65,32,32,27,87,60,13,42,51,77,35)
> x=data.frame(x1,x2)
> x
   x1 x2
1  10 20
2  13 65
3  45 32
4  26 32
5  23 27
6  12 87
7  24 60
8  78 13
9  23 42
10 43 51
11 31 77
12 56 35
> 
```



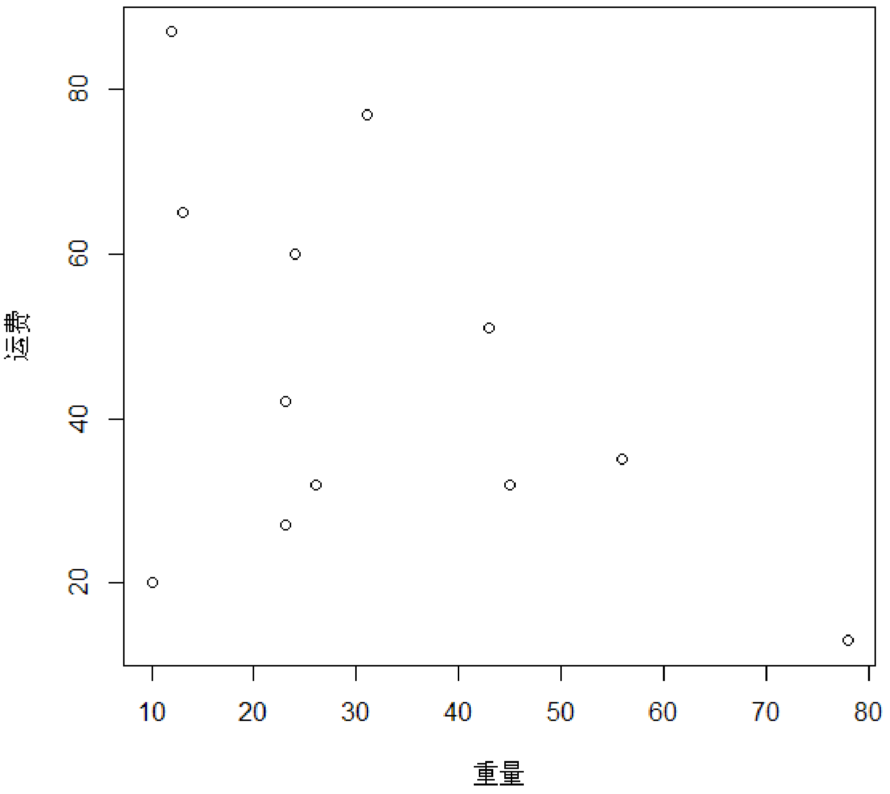
```
> (x=data.frame('重量'=x1,'运费'=x2))
   重量 运费
1    10   20
2    13   65
3    45   32
4    26   32
5    23   27
6    12   87
7    24   60
8    78   13
9    23   42
10   43   51
11   31   77
12   56   35
> 
```

画散点图

- 函数plot( )

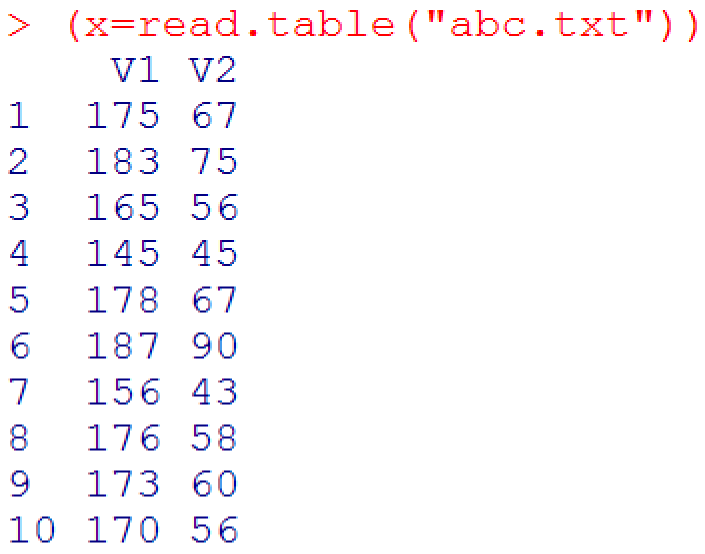


```
> plot(x)
> 
```



读文本文件数据

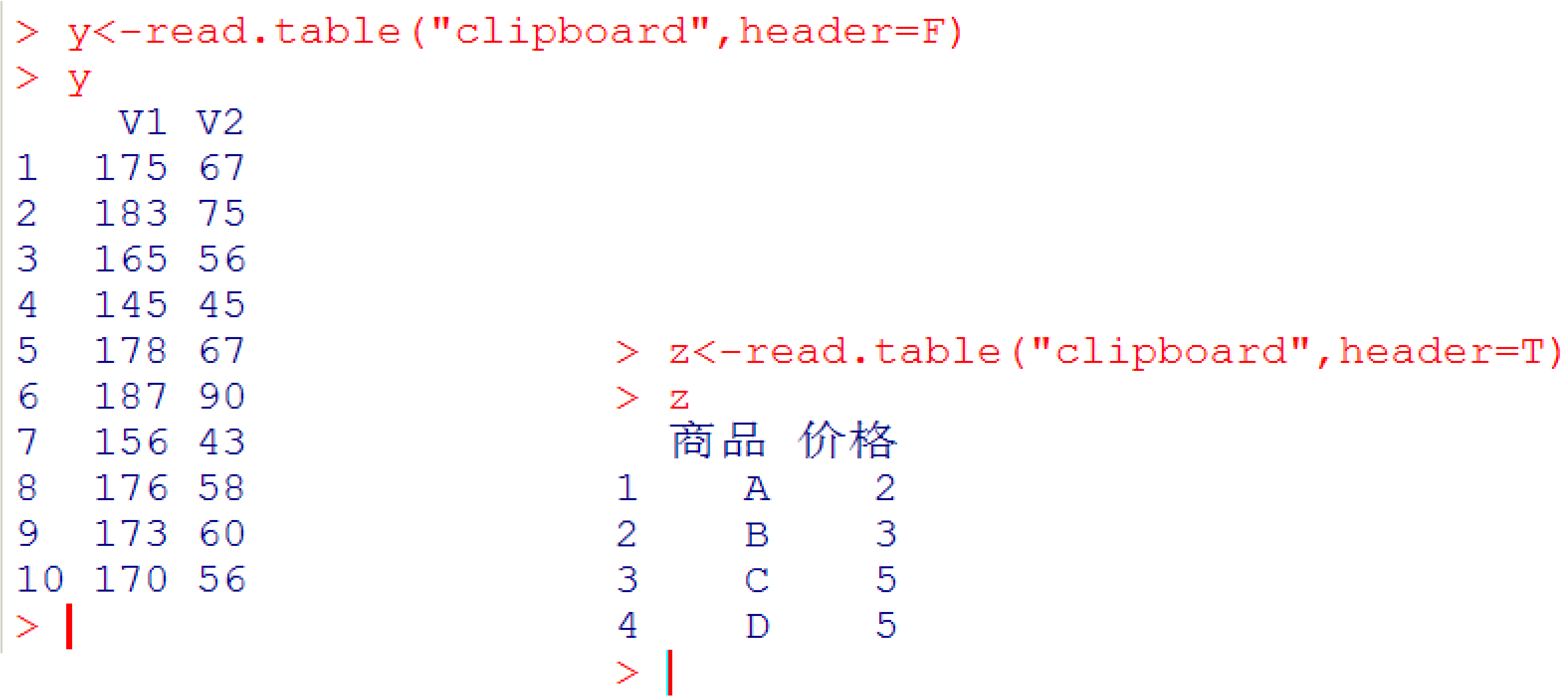
- 先设置工作目录，把文本文件放于该目录下



```
> (x=read.table("abc.txt"))
    V1 V2
1  175 67
2  183 75
3  165 56
4  145 45
5  178 67
6  187 90
7  156 43
8  176 58
9  173 60
10 170 56
```

读剪贴板

- 文本或excel的数据均可通过剪贴板操作



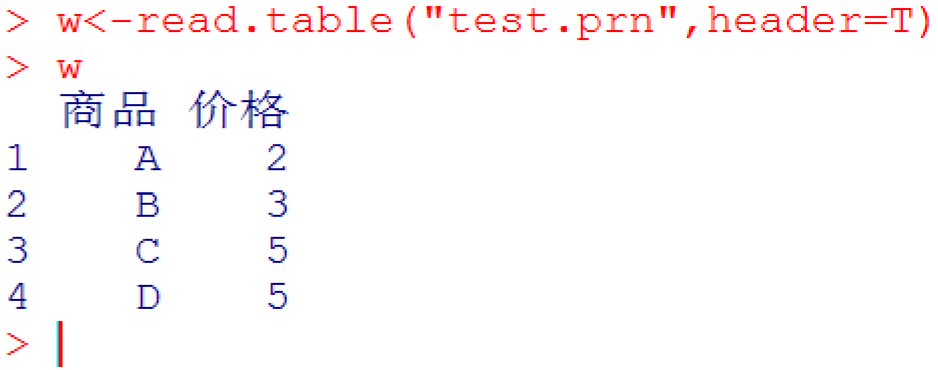
```
> y<-read.table("clipboard",header=F)
> y
    V1 V2
1  175 67
2  183 75
3  165 56
4  145 45
5  178 67
6  187 90
7  156 43
8  176 58
9  173 60
10 170 56
> 
```

```
> z<-read.table("clipboard",header=T)
> z
  商品 价格
1    A    2
2    B    3
3    C    5
4    D    5
> 
```

2015.3.10

读excel文件数据

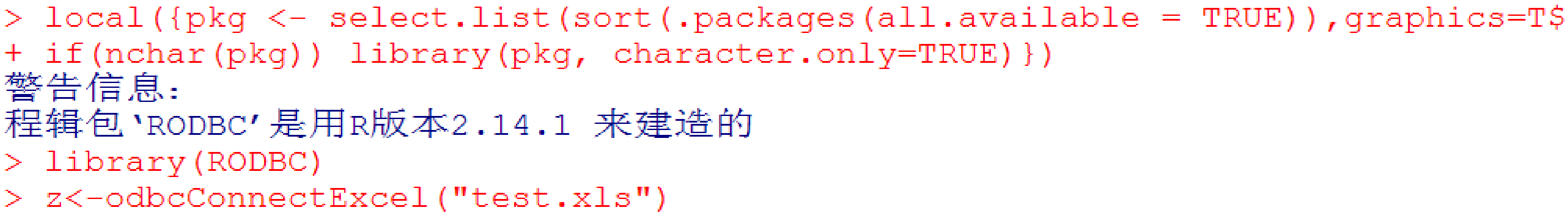
- 方法1：先把excel另存为空格分隔的prn文本格式再读



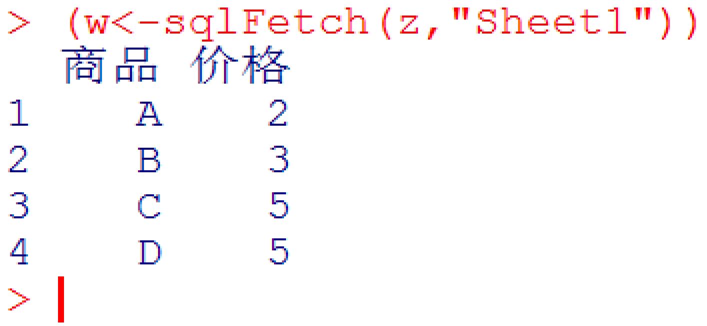
```
> w<-read.table("test.prn",header=T)
> w
  商品 价格
1    A    2
2    B    3
3    C    5
4    D    5
> 
```

读Excel文件数据

- 方法2：安装RODBC包，再通过ODBC读

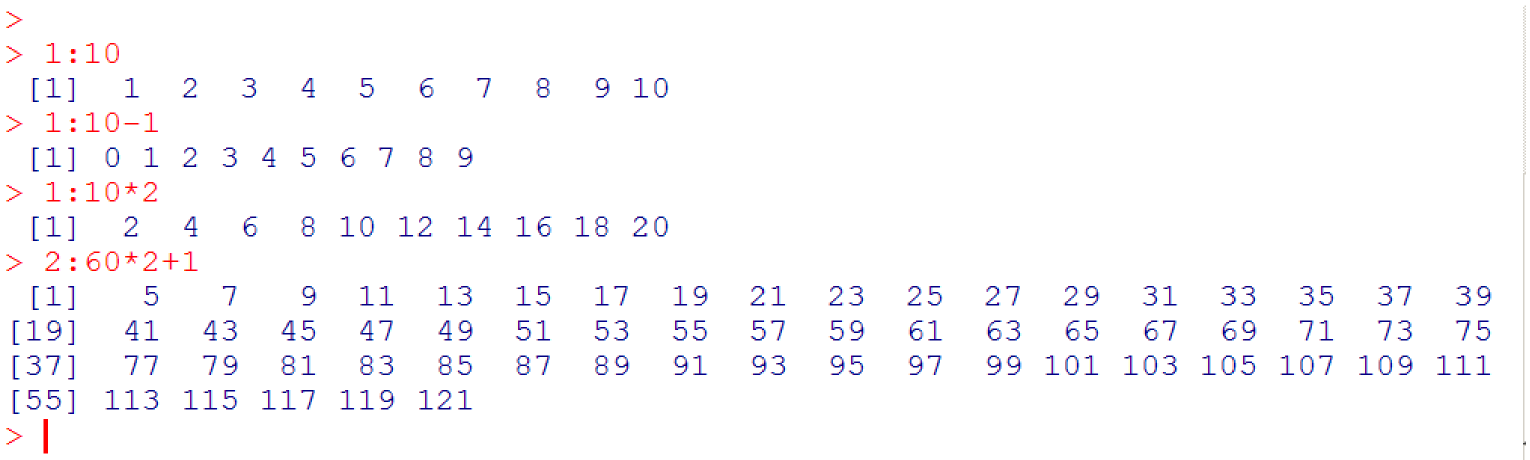


```
> local({pkg <- select.list(sort(.packages(all.available = TRUE)),graphics=T$
+ if(nchar(pkg)) library(pkg, character.only=TRUE)})
警告信息:
程辑包'RODBC'是用R版本2.14.1 来建造的
> library(RODBC)
> z<-odbcConnectExcel("test.xls")
```

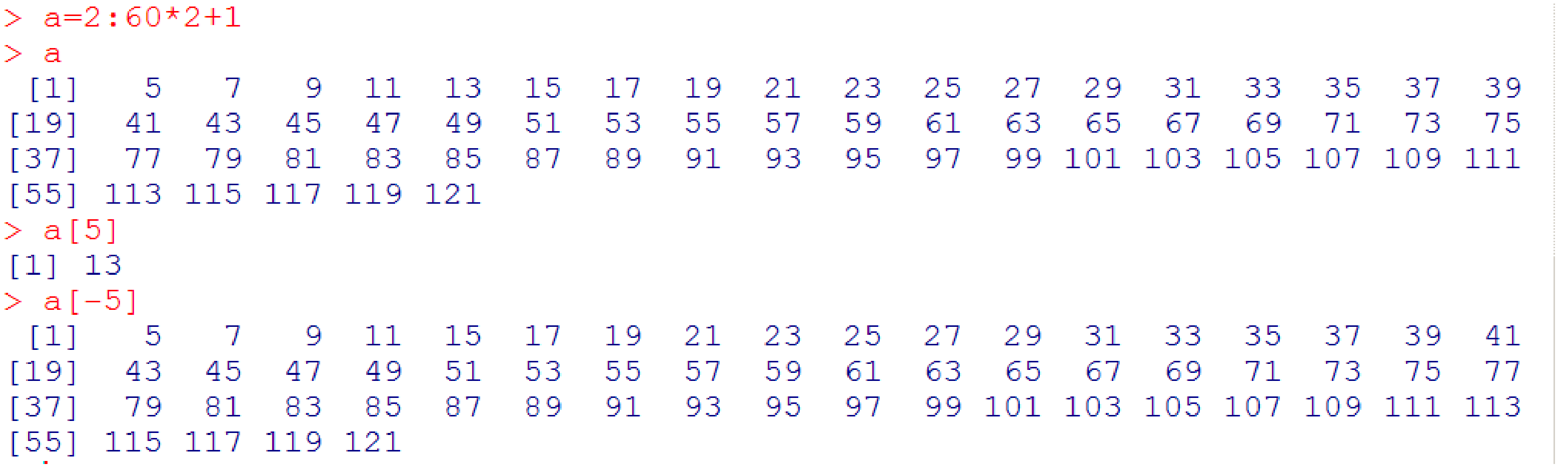


```
> (w<-sqlFetch(z,"Sheet1"))
  商品 价格
1     A     2
2     B     3
3     C     5
4     D     5
> 
```

产生向量

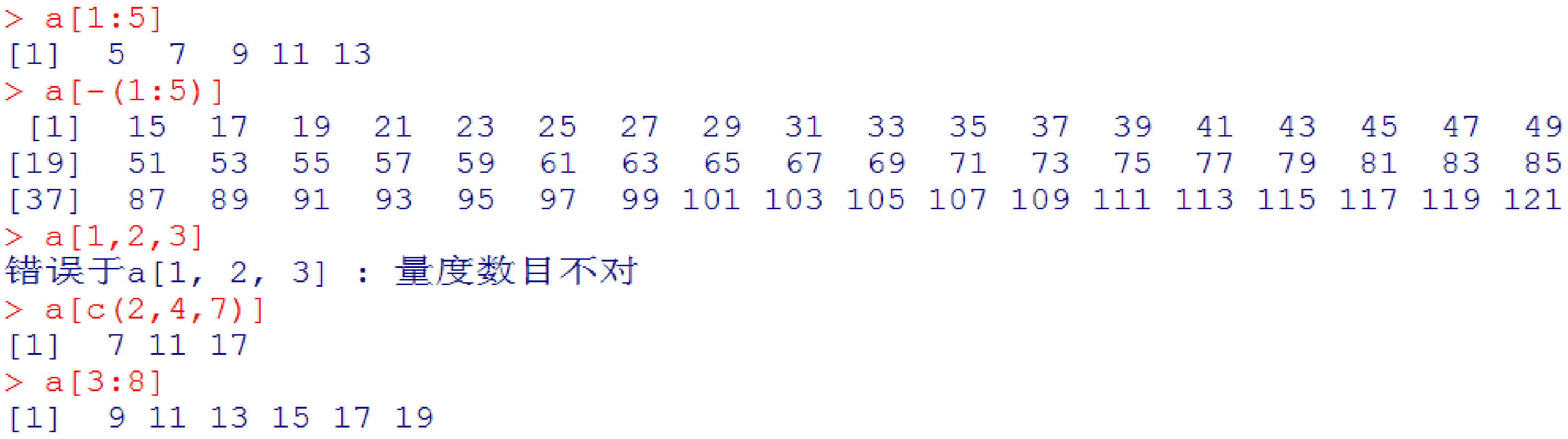


```
>
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> 1:10-1
 [1] 0 1 2 3 4 5 6 7 8 9
> 1:10*2
 [1]  2  4  6  8 10 12 14 16 18 20
> 2:60*2+1
 [1]   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39
[19]  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75
[37]  77  79  81  83  85  87  89  91  93  95  97  99 101 103 105 107 109 111
[55] 113 115 117 119 121
> 
```

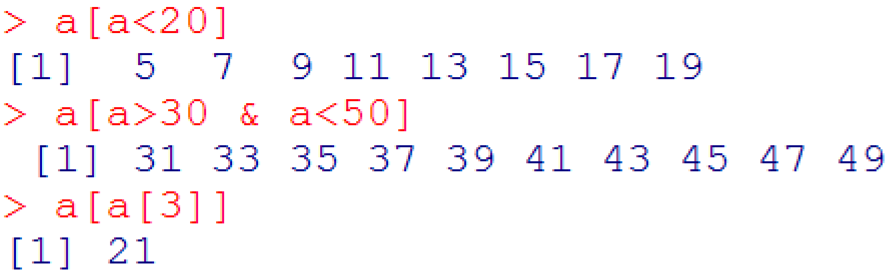


```
> a=2:60*2+1
> a
 [1]   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39
[19]  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75
[37]  77  79  81  83  85  87  89  91  93  95  97  99 101 103 105 107 109 111
[55] 113 115 117 119 121
> a[5]
[1] 13
> a[-5]
 [1]   5   7   9  11  15  17  19  21  23  25  27  29  31  33  35  37  39  41
[19]  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75  77
[37]  79  81  83  85  87  89  91  93  95  97  99 101 103 105 107 109 111 113
[55] 115 117 119 121
```

产生向量



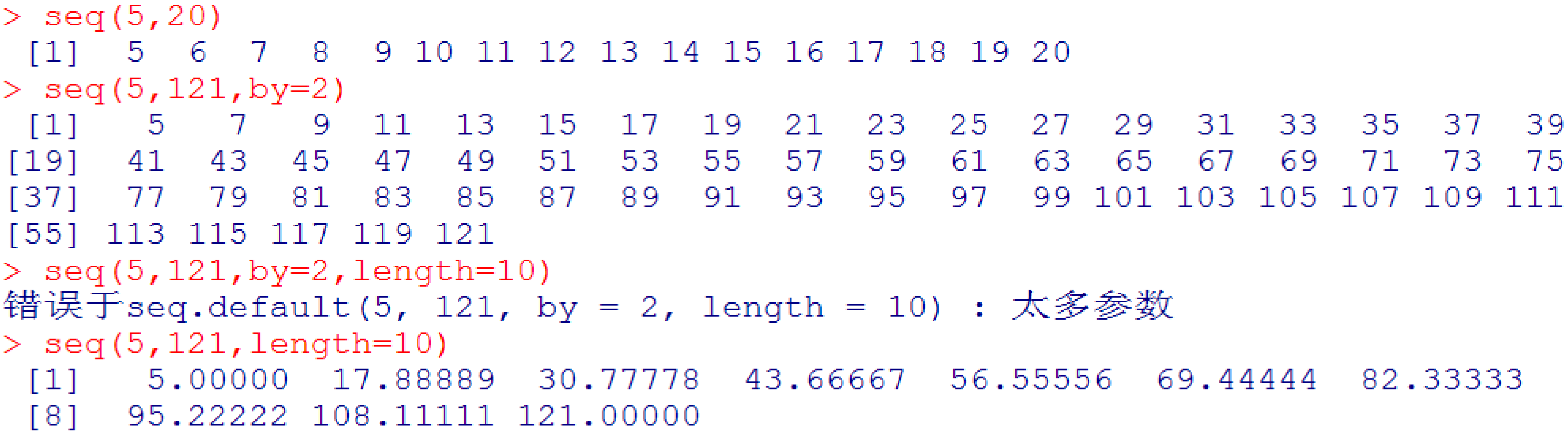
```
> a[1:5]
[1]  5  7  9 11 13
> a[-(1:5)]
 [1]  15  17  19  21  23  25  27  29  31  33  35  37  39  41  43  45  47  49
[19]  51  53  55  57  59  61  63  65  67  69  71  73  75  77  79  81  83  85
[37]  87  89  91  93  95  97  99 101 103 105 107 109 111 113 115 117 119 121
> a[1,2,3]
错误于a[1, 2, 3] : 量度数目不对
> a[c(2,4,7)]
[1]  7 11 17
> a[3:8]
[1]  9 11 13 15 17 19
```



```
> a[a<20]
[1]  5  7  9 11 13 15 17 19
> a[a>30 & a<50]
 [1] 31 33 35 37 39 41 43 45 47 49
> a[a[3]]
[1] 21
```

产生向量

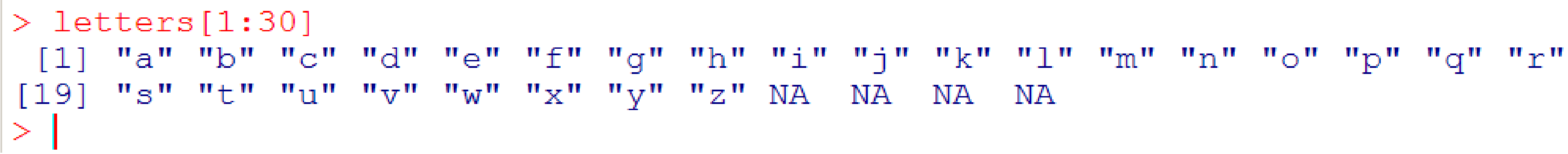
- Seq( )函数



```
> seq(5,20)
 [1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> seq(5,121,by=2)
 [1]   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39
[19]  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75
[37]  77  79  81  83  85  87  89  91  93  95  97  99 101 103 105 107 109 111
[55] 113 115 117 119 121
> seq(5,121,by=2,length=10)
错误于seq.default(5, 121, by = 2, length = 10) : 太多参数
> seq(5,121,length=10)
 [1]   5.00000  17.88889  30.77778  43.66667  56.55556  69.44444  82.33333
 [8]  95.22222 108.11111 121.00000
```

产生向量

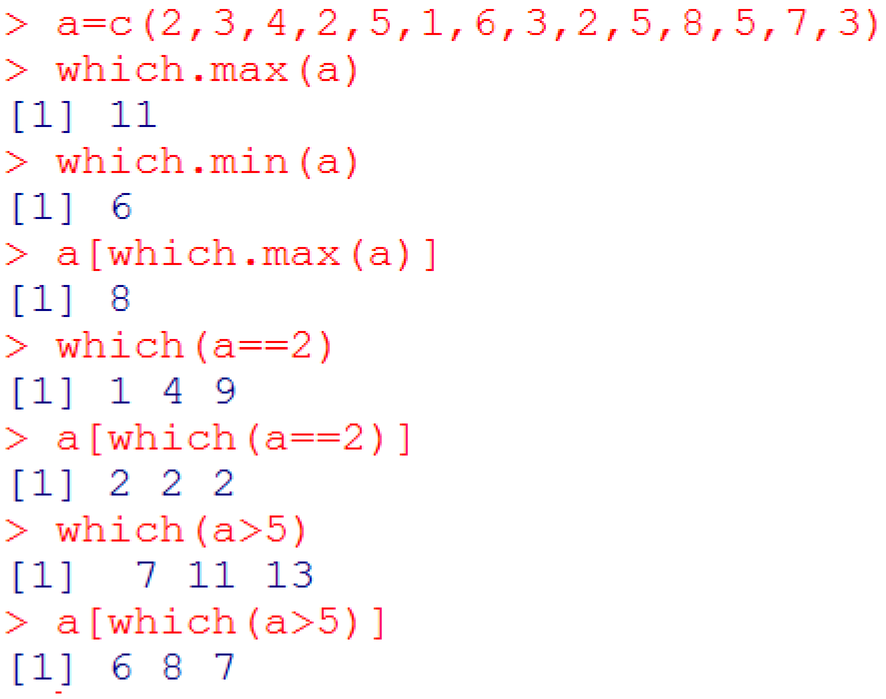
- 产生字母序列 letters



```
> letters[1:30]
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r"
[19] "s" "t" "u" "v" "w" "x" "y" "z" NA  NA  NA  NA
> 
```

新建向量

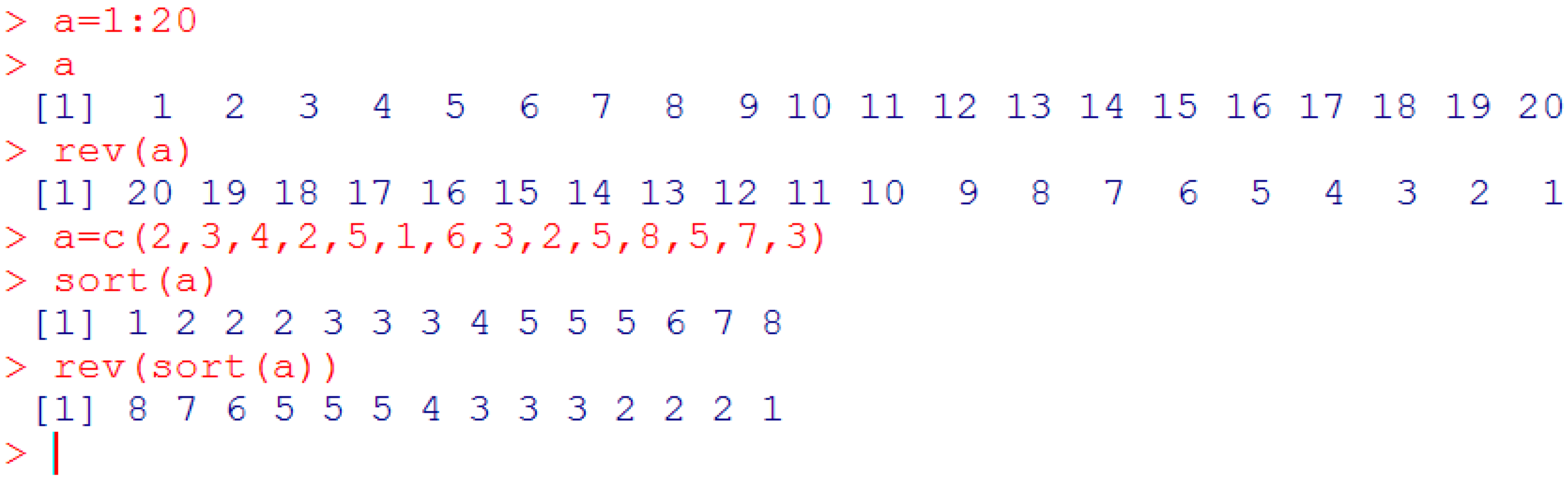
- Which()函数



```
> a=c(2,3,4,2,5,1,6,3,2,5,8,5,7,3)
> which.max(a)
[1] 11
> which.min(a)
[1] 6
> a[which.max(a)]
[1] 8
> which(a==2)
[1] 1 4 9
> a[which(a==2)]
[1] 2 2 2
> which(a>5)
[1]  7 11 13
> a[which(a>5)]
[1] 6 8 7
```

新建向量

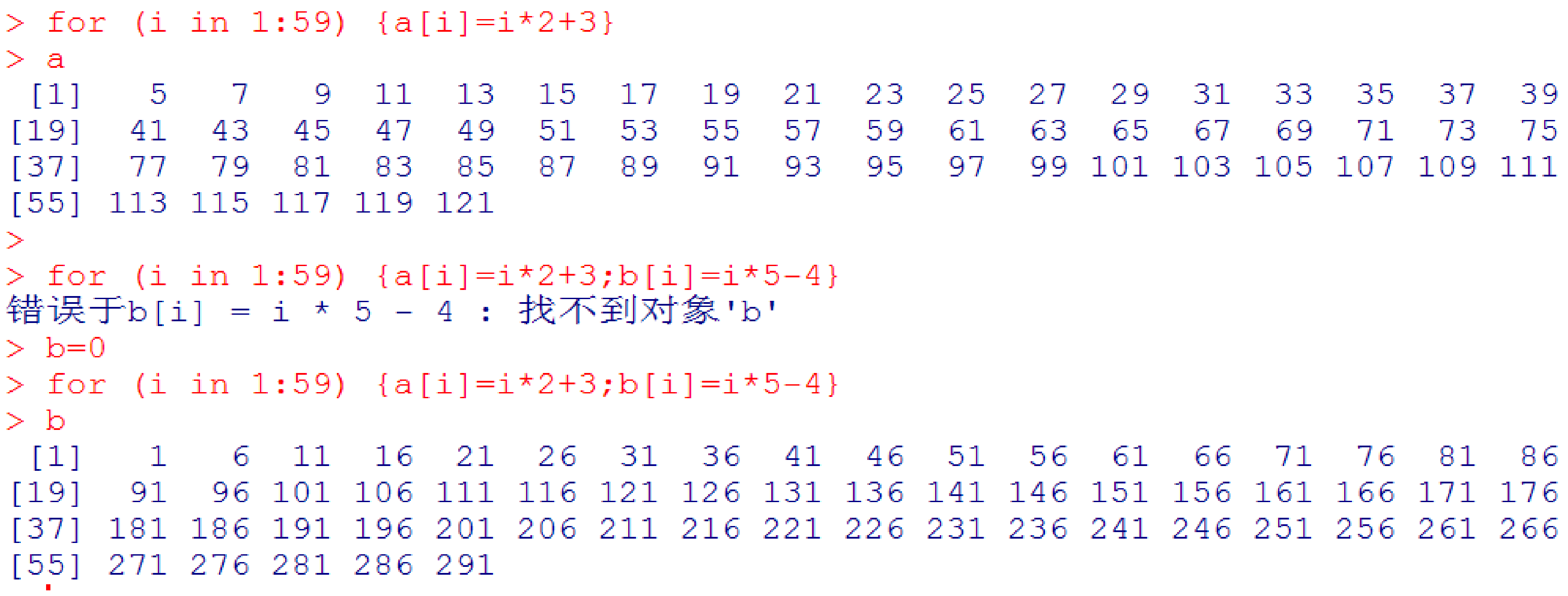
- rev( )函数，sort( )函数



```
> a=1:20
> a
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> rev(a)
 [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
> a=c(2,3,4,2,5,1,6,3,2,5,8,5,7,3)
> sort(a)
 [1] 1 2 2 2 3 3 3 4 5 5 5 6 7 8
> rev(sort(a))
 [1] 8 7 6 5 5 5 4 3 3 3 2 2 2 1
> 
```

循环语句

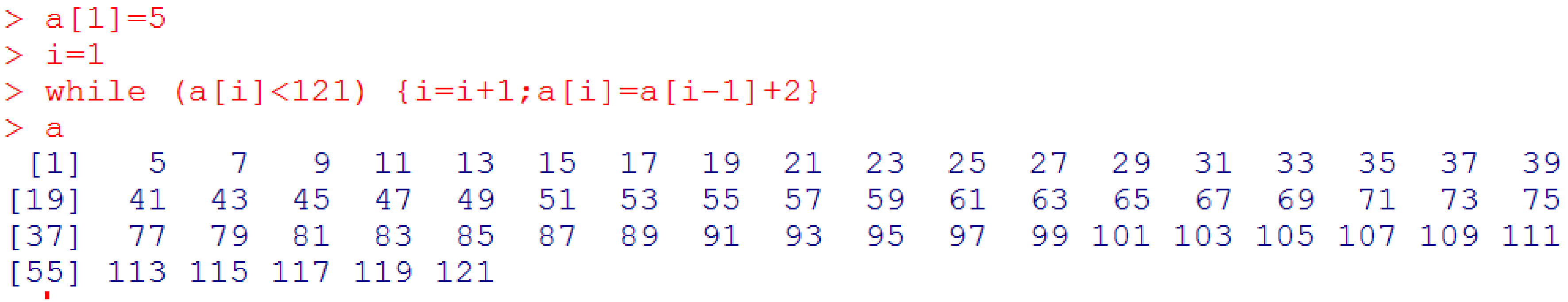
- for语句



```
> for (i in 1:59) {a[i]=i*2+3}
> a
 [1]   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39
[19]  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75
[37]  77  79  81  83  85  87  89  91  93  95  97  99 101 103 105 107 109 111
[55] 113 115 117 119 121
>
> for (i in 1:59) {a[i]=i*2+3;b[i]=i*5-4}
错误于b[i] = i * 5 - 4 : 找不到对象'b'
> b=0
> for (i in 1:59) {a[i]=i*2+3;b[i]=i*5-4}
> b
 [1]   1   6  11  16  21  26  31  36  41  46  51  56  61  66  71  76  81  86
[19]  91  96 101 106 111 116 121 126 131 136 141 146 151 156 161 166 171 176
[37] 181 186 191 196 201 206 211 216 221 226 231 236 241 246 251 256 261 266
[55] 271 276 281 286 291
```

循环语句

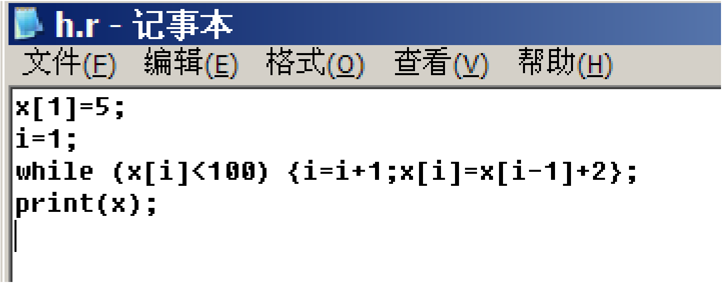
- while语句



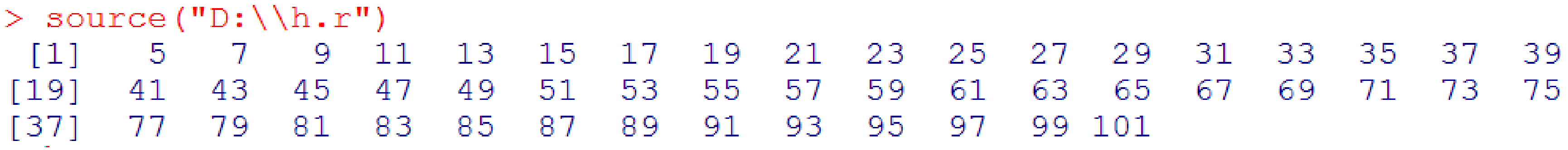
```
> a[1]=5
> i=1
> while (a[i]<121) {i=i+1;a[i]=a[i-1]+2}
> a
 [1]   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39
[19]  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75
[37]  77  79  81  83  85  87  89  91  93  95  97  99 101 103 105 107 109 111
[55] 113 115 117 119 121
```

R脚本

- source( )函数
- print( )函数



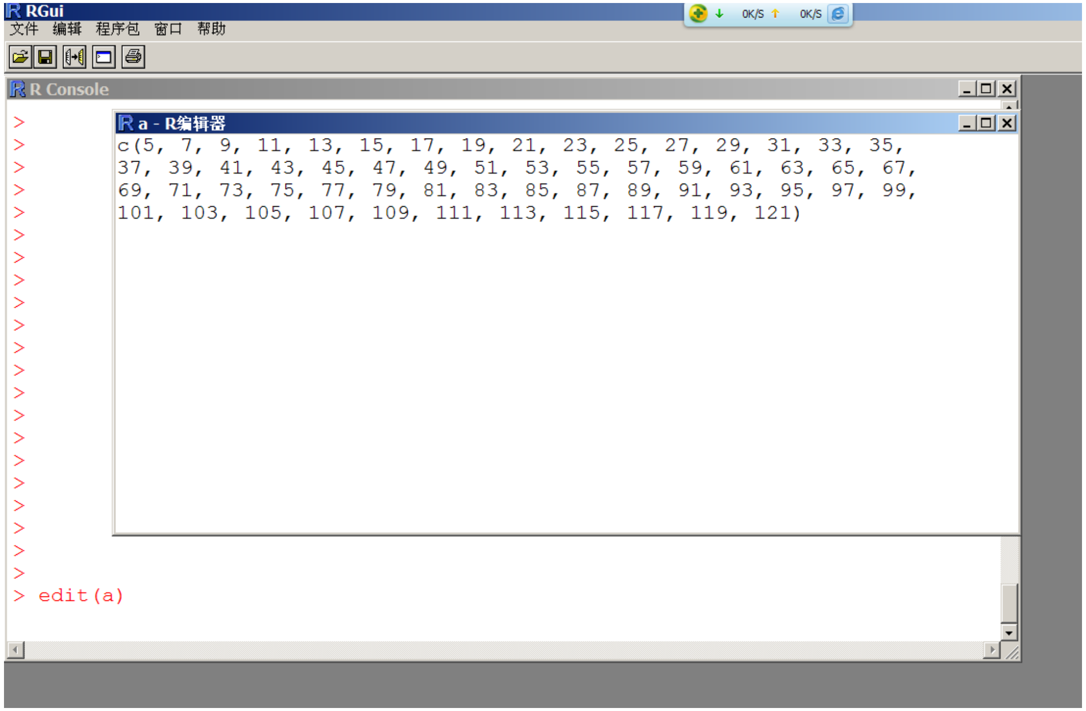
```
x[1]=5;
i=1;
while (x[i]<100) {i=i+1;x[i]=x[i-1]+2};
print(x);
```



```
> source("D:\\h.r")
 [1]   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39
[19]  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75
[37]  77  79  81  83  85  87  89  91  93  95  97  99 101
```

编辑文本对象

- edit( )函数



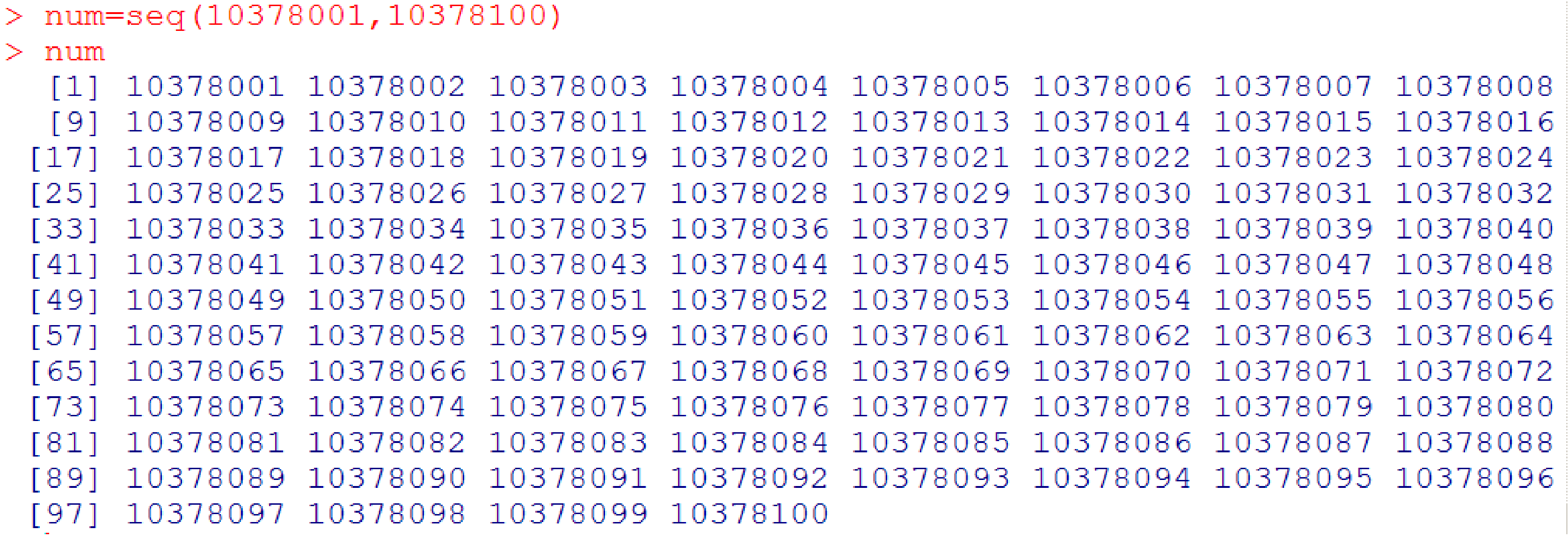
```
c(5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35,
37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67,
69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99,
101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121)
```

```
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
> edit(a)
```

```
> a=edit(a)
> a
 [1] 100   7   9  11  13  15  17  19  21  23  25  27  29  31  33  35  37  39
[19]  41  43  45  47  49  51  53  55  57  59  61  63  65  67  69  71  73  75
[37]  77  79  81  83  85  87  89  91  93  95  97  99 101 103 105 107 109 111
[55] 113 115 117 119 121
```

综合性例子

- 模拟产生统计专业同学的名单（学号区分），记录数学分析，线性代数，概率统计三科成绩，然后进行一些统计分析



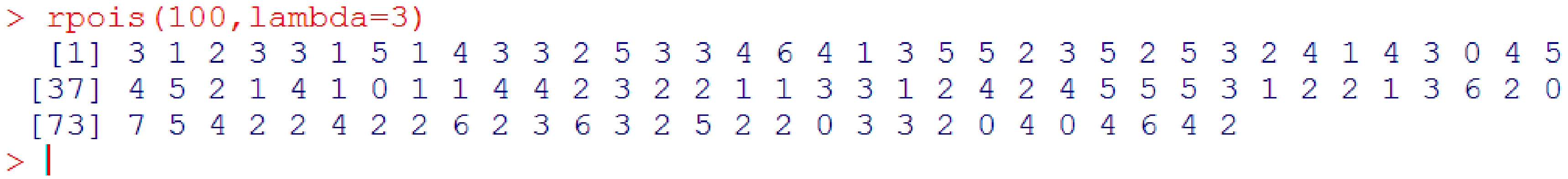
```
> num=seq(10378001,10378100)
> num
  [1] 10378001 10378002 10378003 10378004 10378005 10378006 10378007 10378008
  [9] 10378009 10378010 10378011 10378012 10378013 10378014 10378015 10378016
 [17] 10378017 10378018 10378019 10378020 10378021 10378022 10378023 10378024
 [25] 10378025 10378026 10378027 10378028 10378029 10378030 10378031 10378032
 [33] 10378033 10378034 10378035 10378036 10378037 10378038 10378039 10378040
 [41] 10378041 10378042 10378043 10378044 10378045 10378046 10378047 10378048
 [49] 10378049 10378050 10378051 10378052 10378053 10378054 10378055 10378056
 [57] 10378057 10378058 10378059 10378060 10378061 10378062 10378063 10378064
 [65] 10378065 10378066 10378067 10378068 10378069 10378070 10378071 10378072
 [73] 10378073 10378074 10378075 10378076 10378077 10378078 10378079 10378080
 [81] 10378081 10378082 10378083 10378084 10378085 10378086 10378087 10378088
 [89] 10378089 10378090 10378091 10378092 10378093 10378094 10378095 10378096
 [97] 10378097 10378098 10378099 10378100
```

分布函数

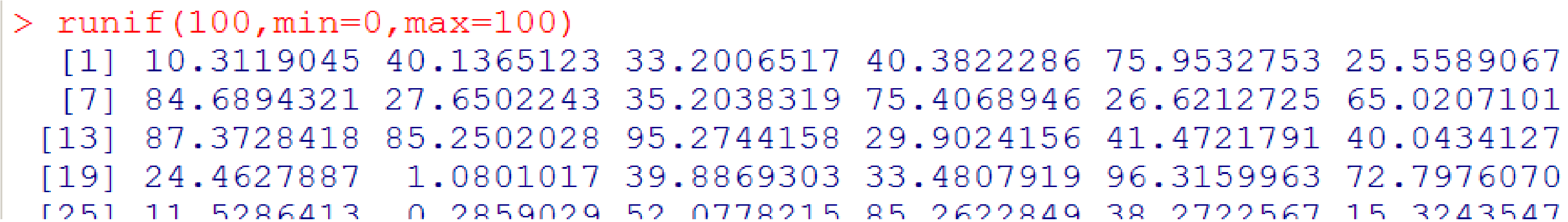
- 正态分布函数rnorm( )
- 泊松分布函数rpois( )
- 指数分布函数rexp( )
- Gamma分布函数rgamma( )
- 均匀分布函数runif( )
- 二项分布函数rbinom( )
- 几何分布函数rgeom( )

泊松分布、均匀分布

$$P(x=k)=\frac{\lambda^k}{k!}e^{-\lambda}$$



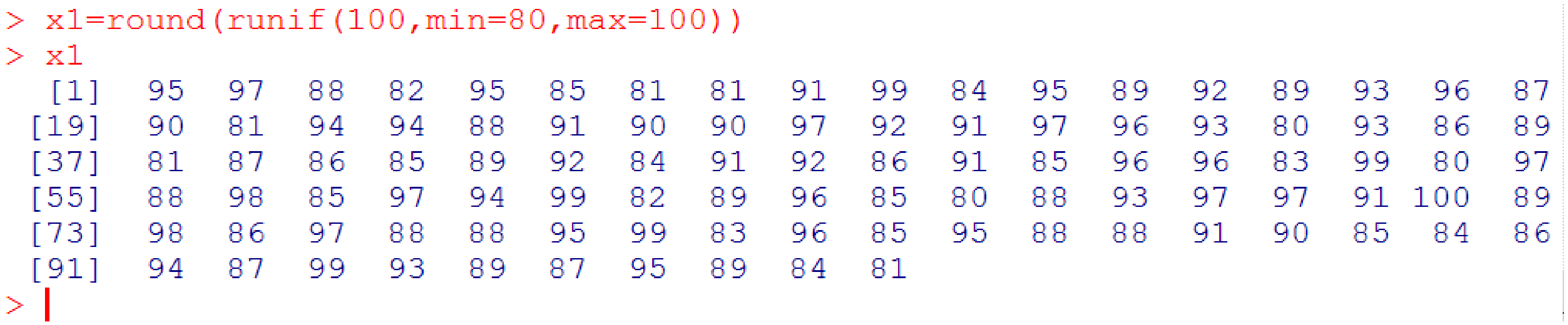
```
> rpois(100,lambda=3)
  [1] 3 1 2 3 3 1 5 1 4 3 3 2 5 3 3 4 6 4 1 3 5 5 2 3 5 2 5 3 2 4 1 4 3 0 4 5
 [37] 4 5 2 1 4 1 0 1 1 4 4 2 3 2 2 1 1 3 3 1 2 4 2 4 5 5 5 3 1 2 2 1 3 6 2 0
 [73] 7 5 4 2 2 4 2 2 6 2 3 6 3 2 5 2 2 0 3 3 2 0 4 0 4 6 4 2
> |
```



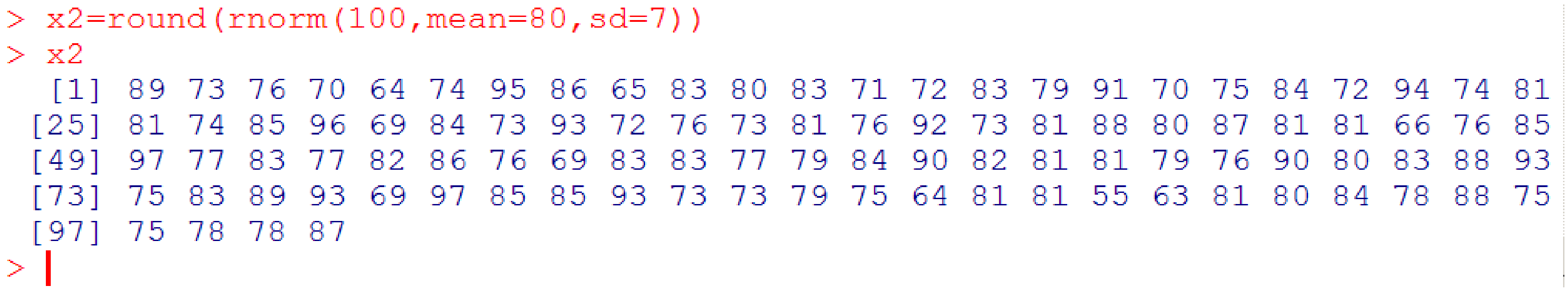
```
> runif(100,min=0,max=100)
  [1] 10.3119045 40.1365123 33.2006517 40.3822286 75.9532753 25.5589067
  [7] 84.6894321 27.6502243 35.2038319 75.4068946 26.6212725 65.0207101
 [13] 87.3728418 85.2502028 95.2744158 29.9024156 41.4721791 40.0434127
 [19] 24.4627887  1.0801017 39.8869303 33.4807919 96.3159963 72.7976070
 [25] 11.5286413  0.2859029 52.0778215 85.2622849 38.2722567 15.3243547
```

模拟成绩

- 用runif和rnorm

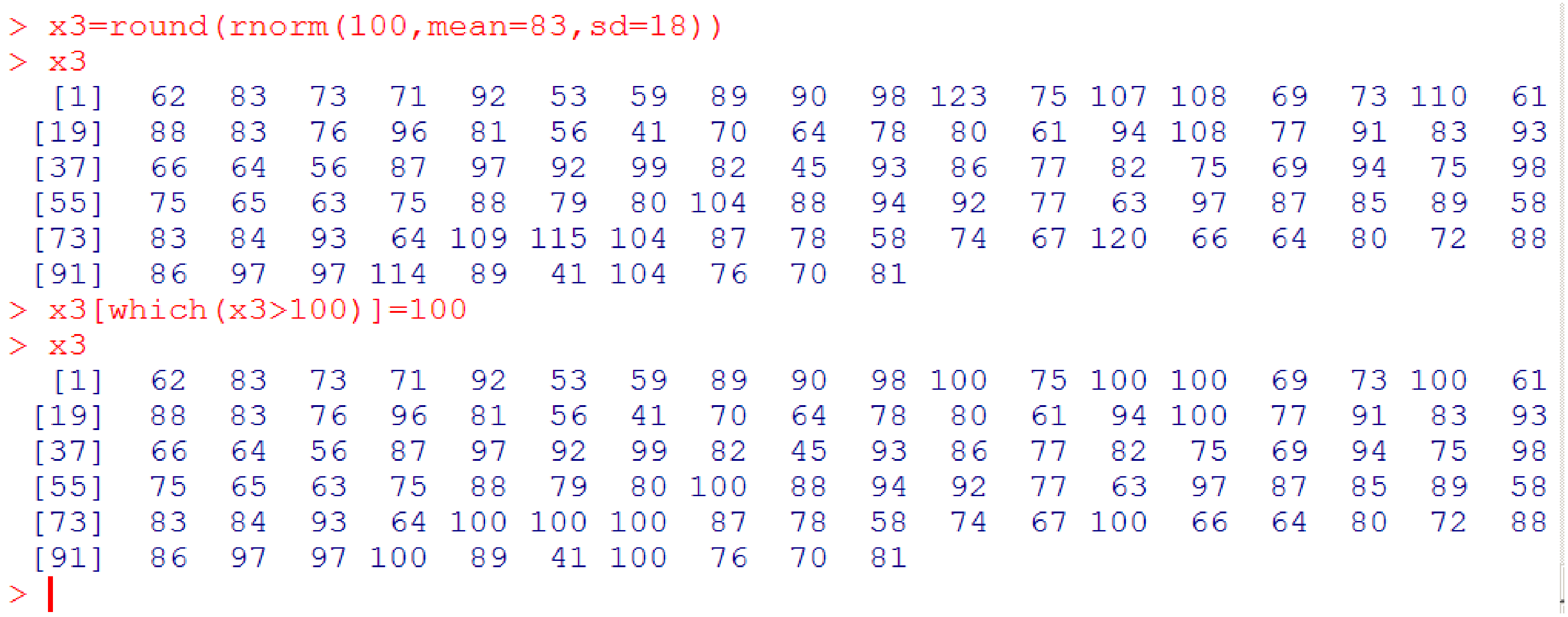


```
> x1=round(runif(100,min=80,max=100))
> x1
  [1]  95  97  88  82  95  85  81  81  91  99  84  95  89  92  89  93  96  87
 [19]  90  81  94  94  88  91  90  90  97  92  91  97  96  93  80  93  86  89
 [37]  81  87  86  85  89  92  84  91  92  86  91  85  96  96  83  99  80  97
 [55]  88  98  85  97  94  99  82  89  96  85  80  88  93  97  97  91 100  89
 [73]  98  86  97  88  88  95  99  83  96  85  95  88  88  91  90  85  84  86
 [91]  94  87  99  93  89  87  95  89  84  81
>
```



```
> x2=round(rnorm(100,mean=80,sd=7))
> x2
  [1] 89 73 76 70 64 74 95 86 65 83 80 83 71 72 83 79 91 70 75 84 72 94 74 81
 [25] 81 74 85 96 69 84 73 93 72 76 73 81 76 92 73 81 88 80 87 81 81 66 76 85
 [49] 97 77 83 77 82 86 76 69 83 83 77 79 84 90 82 81 81 79 76 90 80 83 88 93
 [73] 75 83 89 93 69 97 85 85 93 73 73 79 75 64 81 81 55 63 81 80 84 78 88 75
 [97] 75 78 78 87
>
```

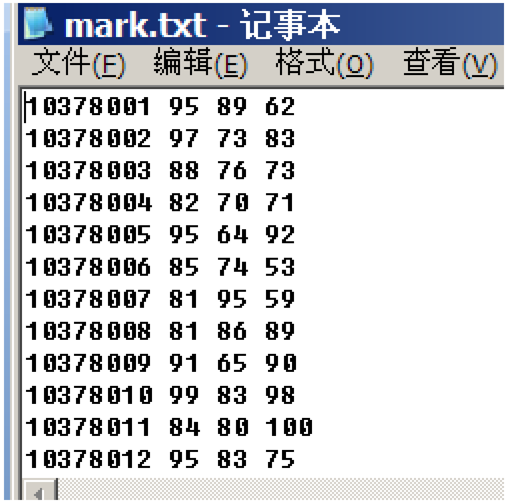
模拟成绩



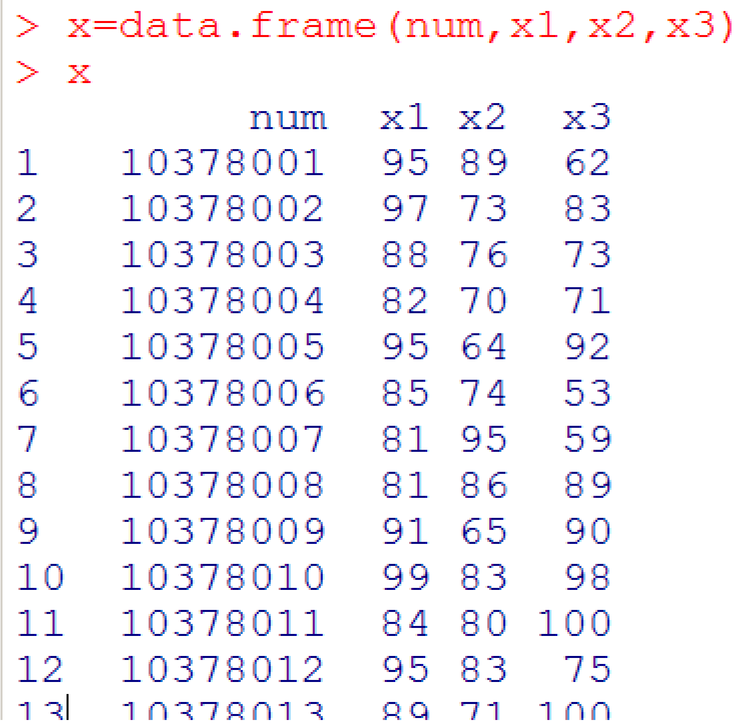
```
> x3=round(rnorm(100,mean=83,sd=18))
> x3
  [1]  62  83  73  71  92  53  59  89  90  98 123  75 107 108  69  73 110  61
 [19]  88  83  76  96  81  56  41  70  64  78  80  61  94 108  77  91  83  93
 [37]  66  64  56  87  97  92  99  82  45  93  86  77  82  75  69  94  75  98
 [55]  75  65  63  75  88  79  80 104  88  94  92  77  63  97  87  85  89  58
 [73]  83  84  93  64 109 115 104  87  78  58  74  67 120  66  64  80  72  88
 [91]  86  97  97 114  89  41 104  76  70  81
> x3[which(x3>100)]=100
> x3
  [1]  62  83  73  71  92  53  59  89  90  98 100  75 100 100  69  73 100  61
 [19]  88  83  76  96  81  56  41  70  64  78  80  61  94 100  77  91  83  93
 [37]  66  64  56  87  97  92  99  82  45  93  86  77  82  75  69  94  75  98
 [55]  75  65  63  75  88  79  80 100  88  94  92  77  63  97  87  85  89  58
 [73]  83  84  93  64 100 100 100  87  78  58  74  67 100  66  64  80  72  88
 [91]  86  97  97 100  89  41 100  76  70  81
> 
```

合成数据框并保存到硬盘

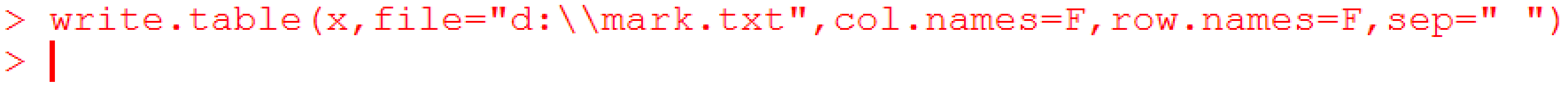
- data.frame( )
- write.table



```
10378001 95 89 62
10378002 97 73 83
10378003 88 76 73
10378004 82 70 71
10378005 95 64 92
10378006 85 74 53
10378007 81 95 59
10378008 81 86 89
10378009 91 65 90
10378010 99 83 98
10378011 84 80 100
10378012 95 83 75
```



```
> x=data.frame(num,x1,x2,x3)
> x
        num x1 x2  x3
1  10378001 95 89  62
2  10378002 97 73  83
3  10378003 88 76  73
4  10378004 82 70  71
5  10378005 95 64  92
6  10378006 85 74  53
7  10378007 81 95  59
8  10378008 81 86  89
9  10378009 91 65  90
10 10378010 99 83  98
11 10378011 84 80 100
12 10378012 95 83  75
13 10378013 89 71 100
```

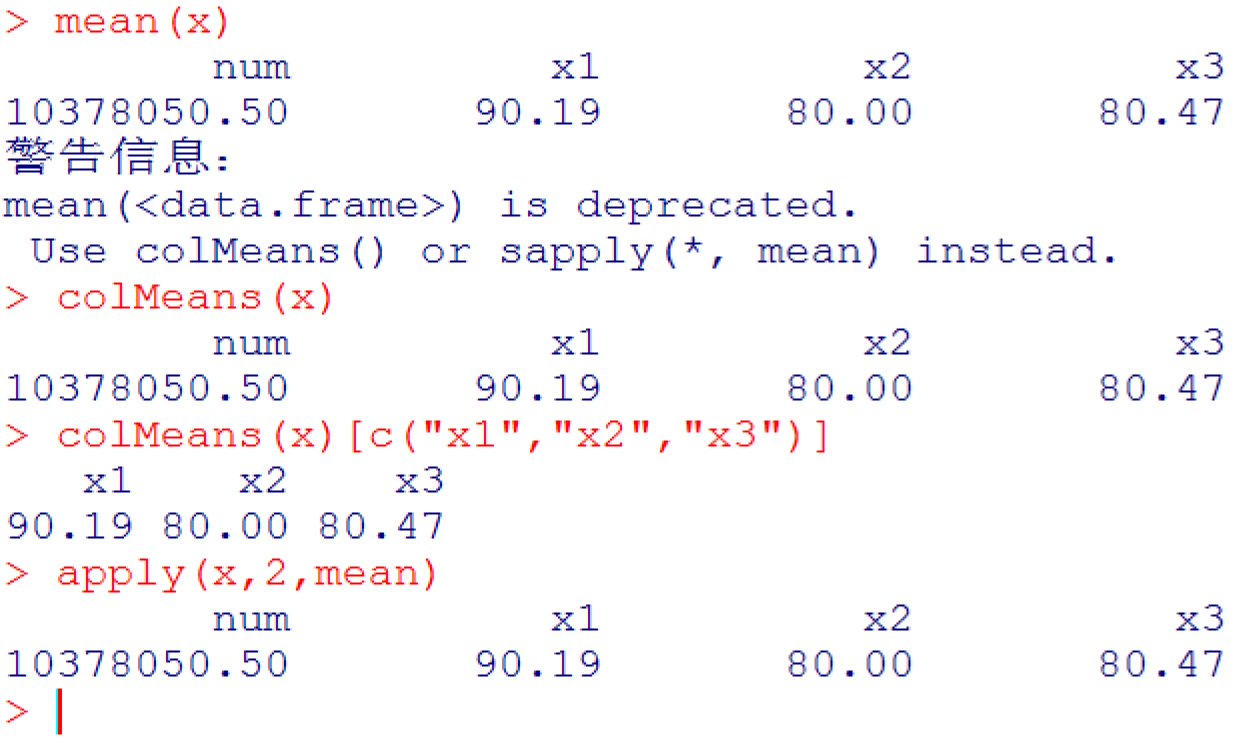


```
> write.table(x,file="d:\\mark.txt",col.names=F,row.names=F,sep=" ")
> 
```

2015.3.10

计算各科平均分

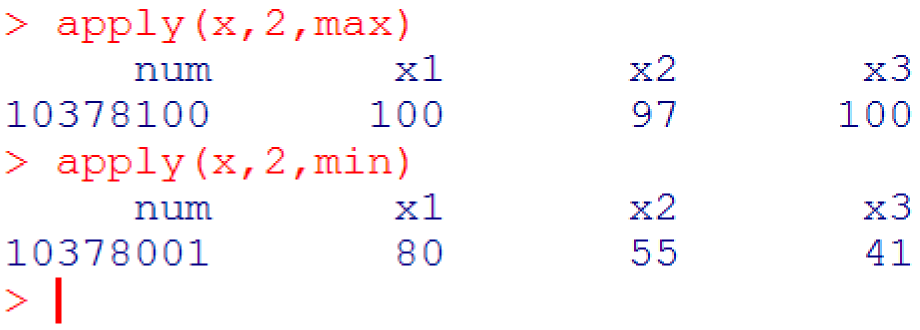
- 函数mean( ), colMeans( ), apply( )



```
> mean(x)
        num          x1          x2          x3 
10378050.50       90.19       80.00       80.47 
警告信息：
mean(<data.frame>) is deprecated.
 Use colMeans() or sapply(*, mean) instead.
> colMeans(x)
        num          x1          x2          x3 
10378050.50       90.19       80.00       80.47 
> colMeans(x)[c("x1","x2","x3")]
   x1    x2    x3 
90.19 80.00 80.47 
> apply(x,2,mean)
        num          x1          x2          x3 
10378050.50       90.19       80.00       80.47 
> 
```

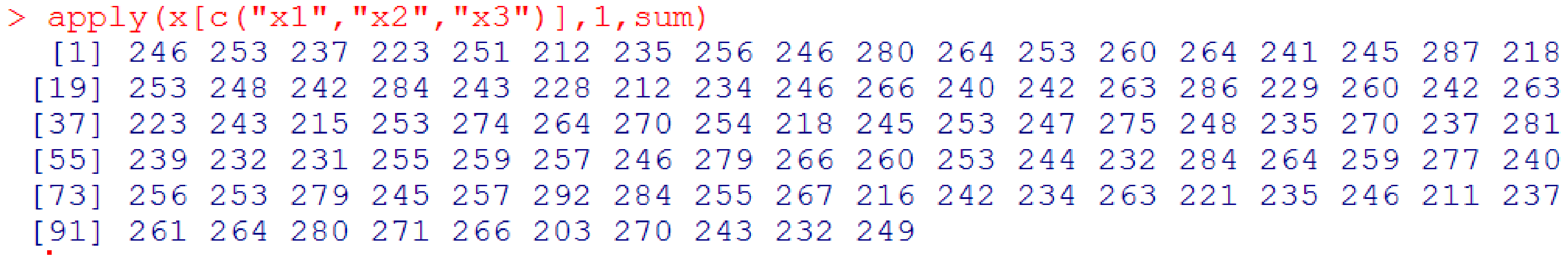
求各科最高最低分

- 函数max( ),min( ),apply( )



```
> apply(x,2,max)
     num       x1       x2       x3
10378100      100       97      100
> apply(x,2,min)
     num       x1       x2       x3
10378001       80       55       41
> 
```

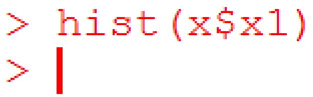
求出每人总分



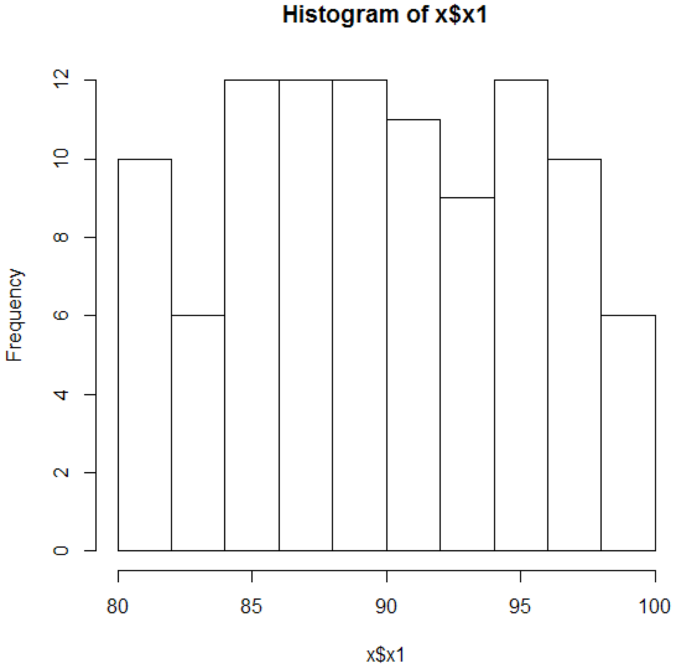
```
> apply(x[c("x1","x2","x3")],1,sum)
  [1] 246 253 237 223 251 212 235 256 246 280 264 253 260 264 241 245 287 218
 [19] 253 248 242 284 243 228 212 234 246 266 240 242 263 286 229 260 242 263
 [37] 223 243 215 253 274 264 270 254 218 245 253 247 275 248 235 270 237 281
 [55] 239 232 231 255 259 257 246 279 266 260 253 244 232 284 264 259 277 240
 [73] 256 253 279 245 257 292 284 255 267 216 242 234 263 221 235 246 211 237
 [91] 261 264 280 271 266 203 270 243 232 249
```

对x1进行直方图分析

- 绘制直方图函数hist( )

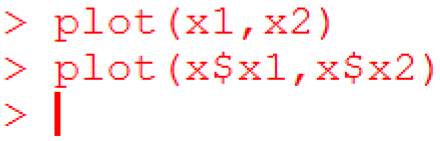


```
> hist(x$x1)
> 
```

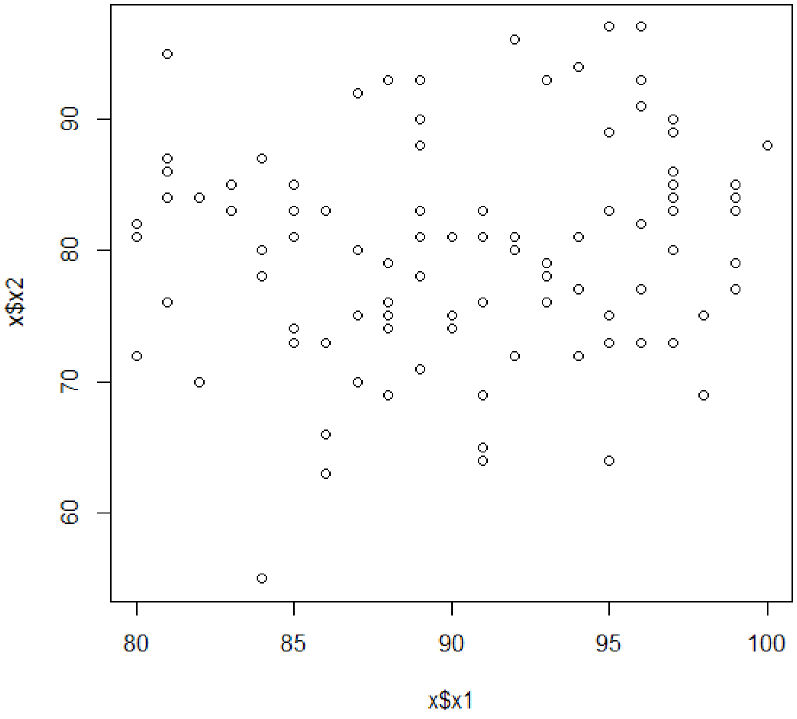


探索各科成绩的关联关系

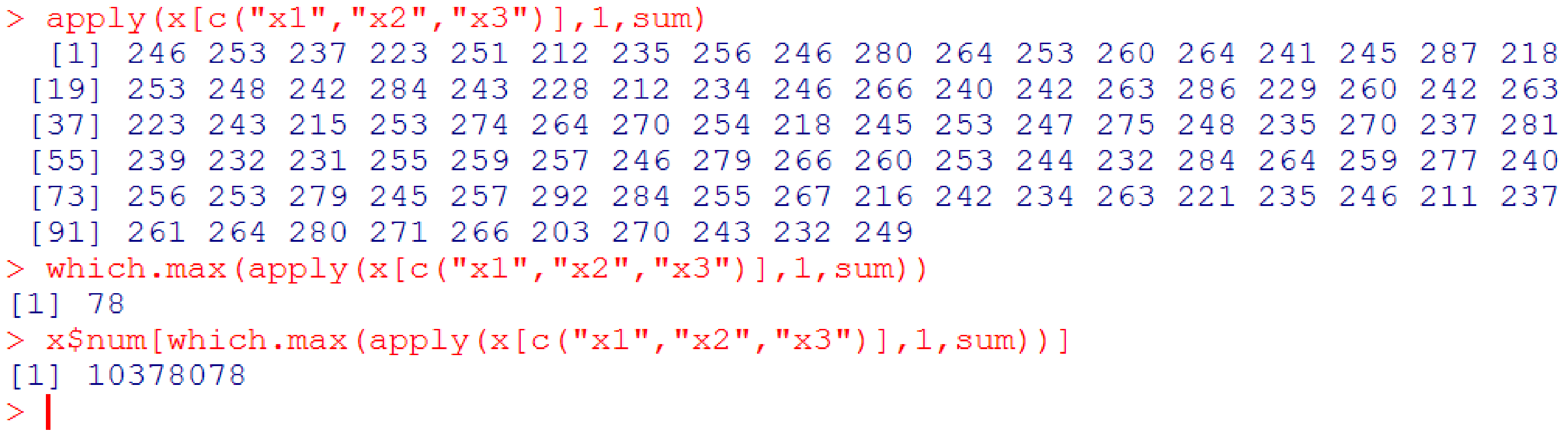
- 散点图绘制函数plot( )



```
> plot(x1,x2)
> plot(x$x1,x$x2)
> 
```



总分最高的同学



```
> apply(x[c("x1","x2","x3")],1,sum)
  [1] 246 253 237 223 251 212 235 256 246 280 264 253 260 264 241 245 287 218
 [19] 253 248 242 284 243 228 212 234 246 266 240 242 263 286 229 260 242 263
 [37] 223 243 215 253 274 264 270 254 218 245 253 247 275 248 235 270 237 281
 [55] 239 232 231 255 259 257 246 279 266 260 253 244 232 284 264 259 277 240
 [73] 256 253 279 245 257 292 284 255 267 216 242 234 263 221 235 246 211 237
 [91] 261 264 280 271 266 203 270 243 232 249
> which.max(apply(x[c("x1","x2","x3")],1,sum))
[1] 78
> x$num[which.max(apply(x[c("x1","x2","x3")],1,sum))]
[1] 10378078
> 
```

关于成绩的一些统计

- 求每个同学的总分

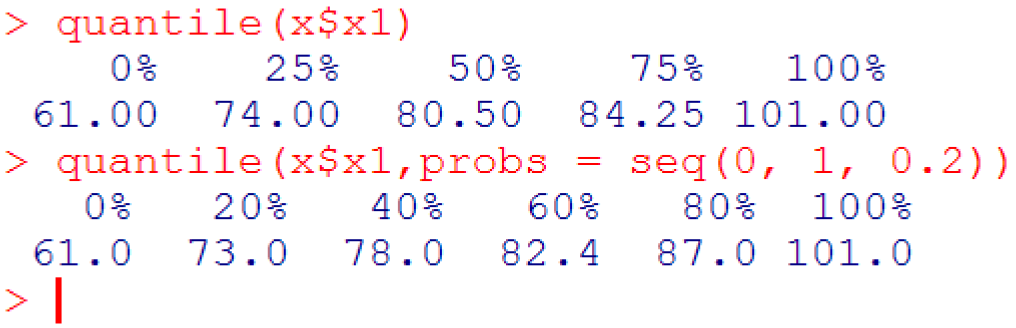
```
data.frame(num,apply(x[2:4],1,sum))
```

- 求数学分析分数最高的三名同学学号

```
num[(sort(x$x1,index.return=T,decreasing=T))$ix[1:3]]
```

常见的数据描述性分析

- 中位数 median()
- 百分位数 quantile( )

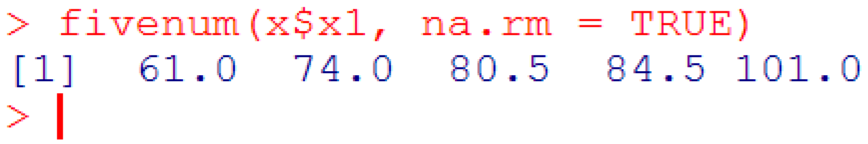


```
> quantile(x$x1)
    0%     25%     50%     75%    100% 
 61.00   74.00   80.50   84.25  101.00 
> quantile(x$x1,probs = seq(0, 1, 0.2))
   0%   20%   40%   60%   80%  100% 
 61.0  73.0  78.0  82.4  87.0 101.0 
> 
```

常见的数据描述性分析

- 五数总括：

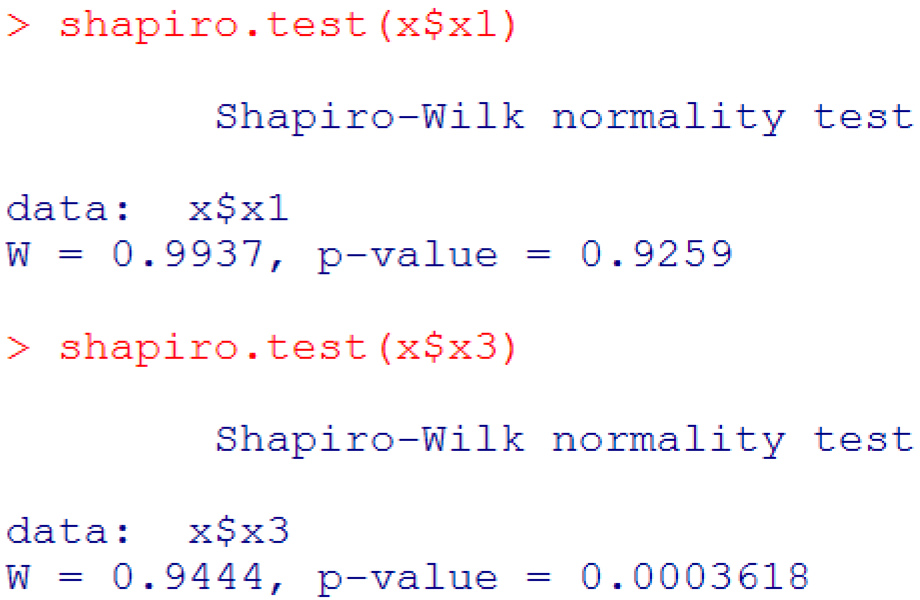
中位数 m_e, 下四分位数 Q_1, 上四分位数 Q_3, 最小值 min 和最大值 max.



```
> fivenum(x$x1, na.rm = TRUE)
[1]  61.0  74.0  80.5  84.5 101.0
> 
```

常见的数据描述性分析

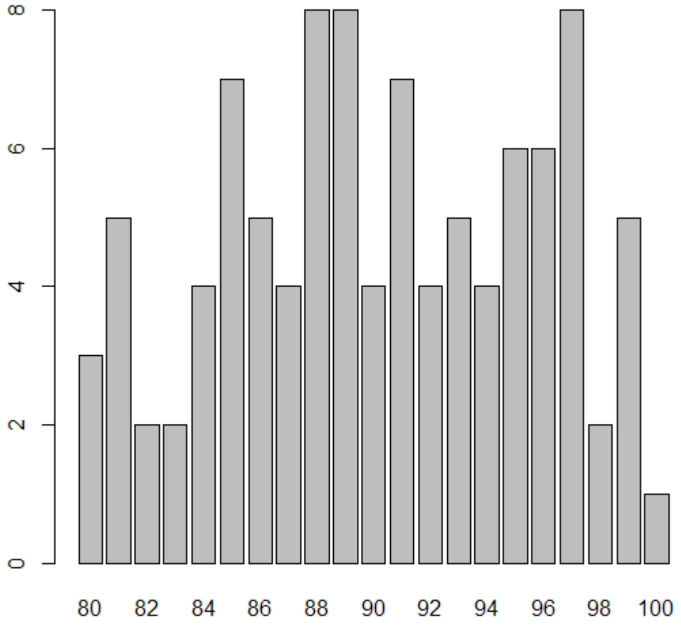
- 正态性检验：函数shapiro.test( )
- P>0.05，正态性分布

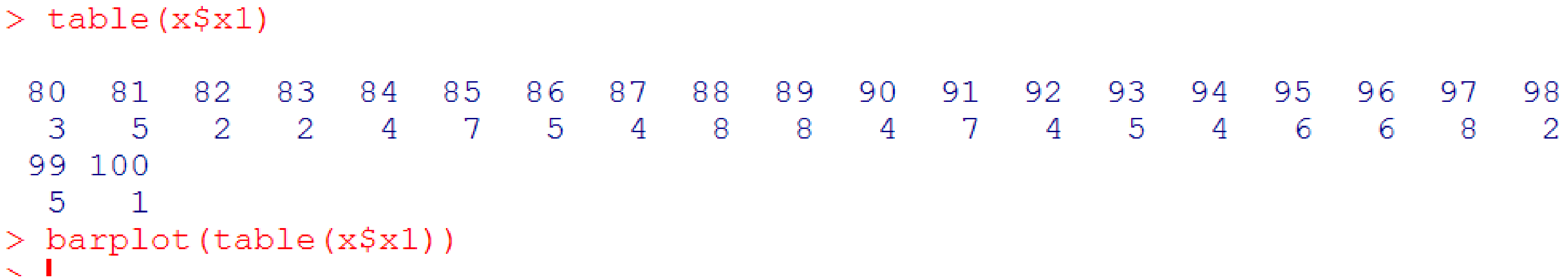


```
> shapiro.test(x$x1)

        Shapiro-Wilk normality test

data:  x$x1
W = 0.9937, p-value = 0.9259

> shapiro.test(x$x3)

        Shapiro-Wilk normality test

data:  x$x3
W = 0.9444, p-value = 0.0003618
```

列联表分析

- 列联函数table( )，柱状图绘制函数barplot( )

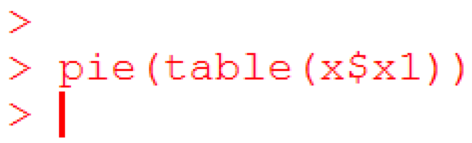




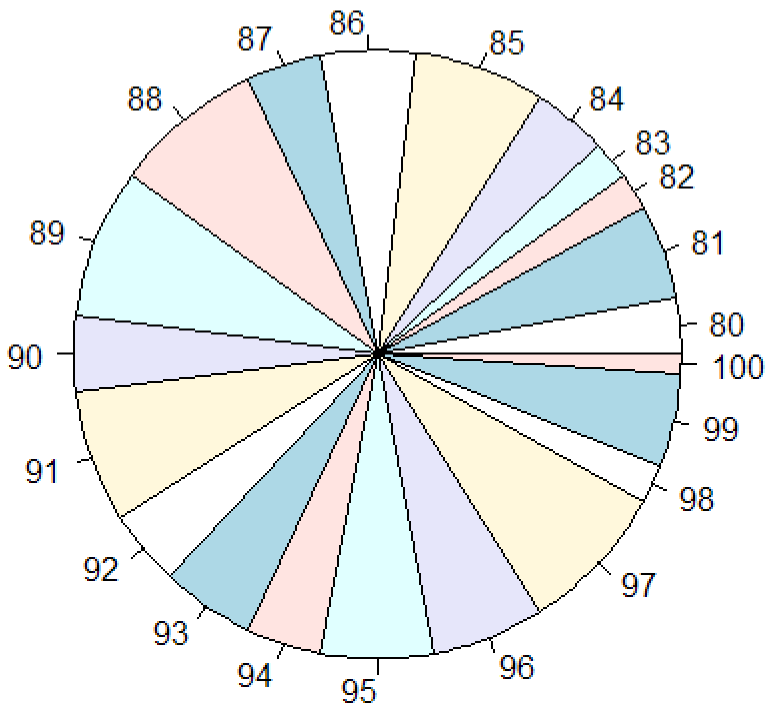
```
> table(x$x1)

 80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
  3   5   2   2   4   7   5   4   8   8   4   7   4   5   4   6   6   8   2
 99 100
  5   1
> barplot(table(x$x1))
>
```

饼图

- 饼图绘制函数pie( )

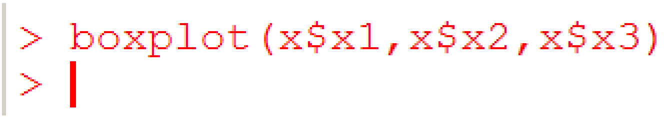


```
>
> pie(table(x$x1))
>
```

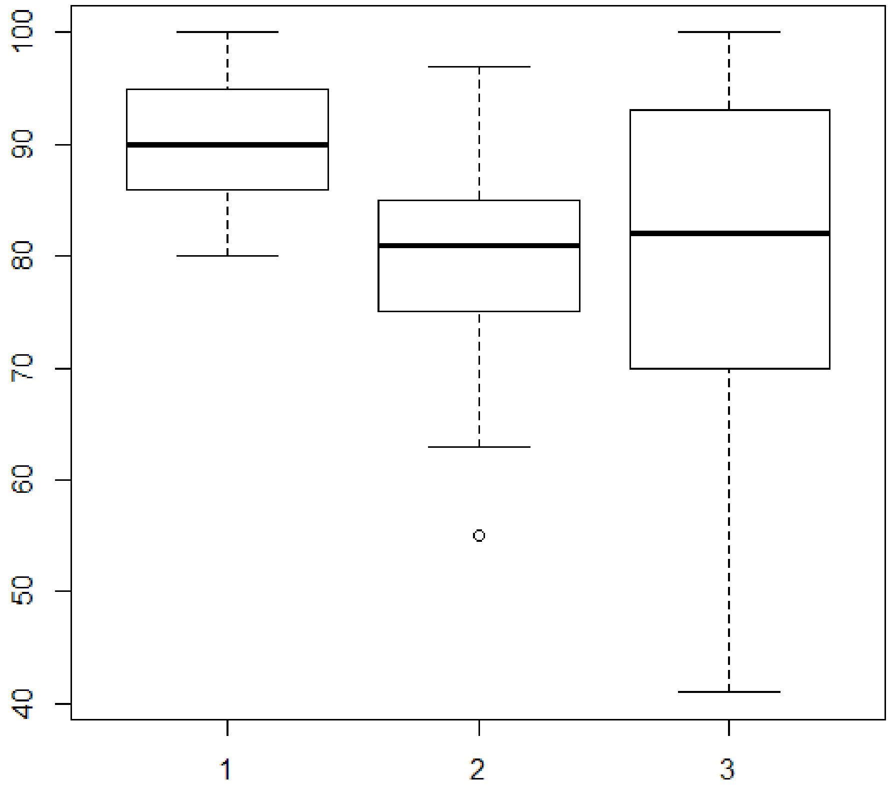


箱尾图

- 箱子的上下横线为样本的25%和75%分位数
- 箱子中间的横线为样本的中位数
- 上下延伸的直线称为尾线，尾线的尽头为最高值和最低值
- 异常值

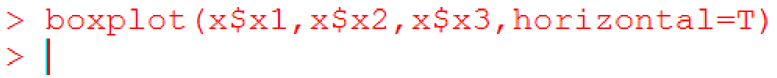


```
> boxplot(x$x1,x$x2,x$x3)
> 
```

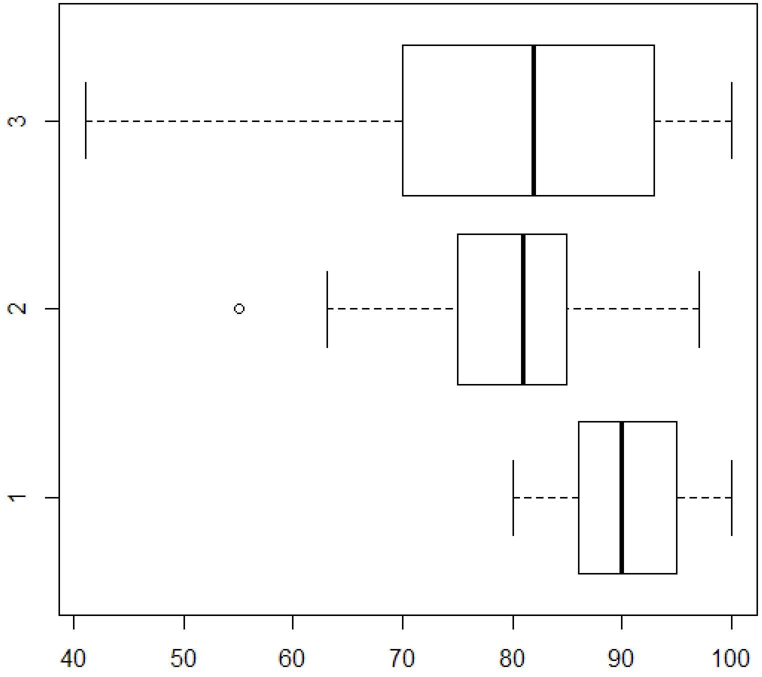


箱尾图

- 水平放置的箱尾图

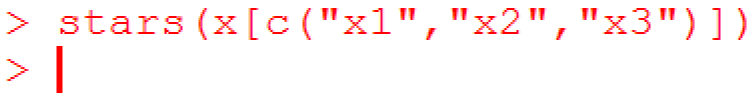


```
> boxplot(x$x1,x$x2,x$x3,horizontal=T)
> |
```

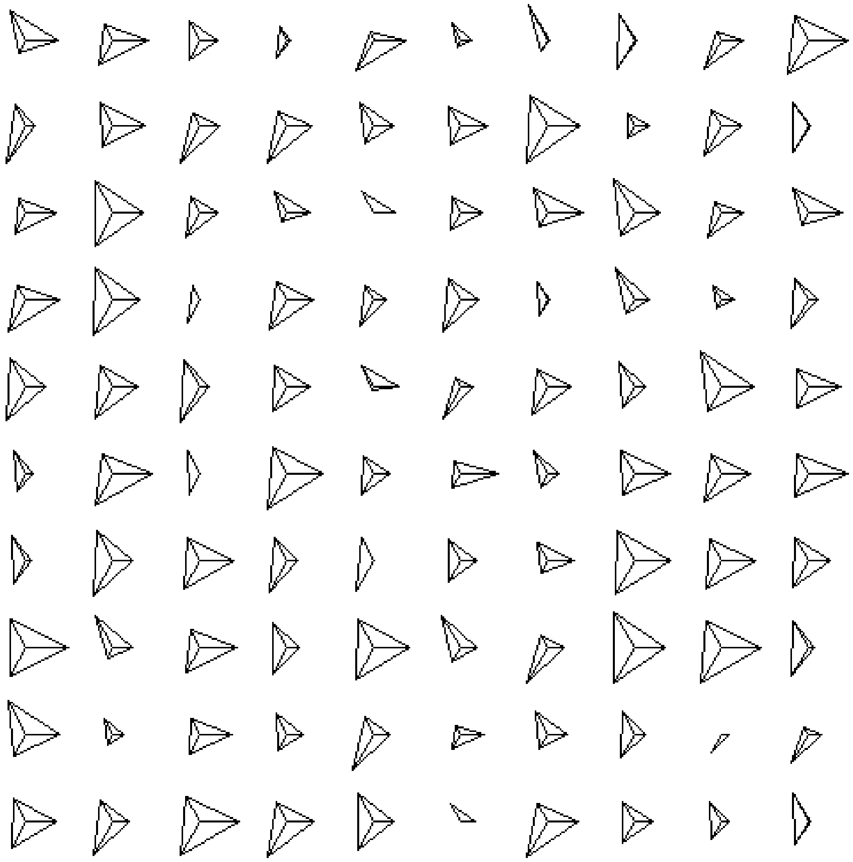


星相图

- 每个观测单位的数值表示为一个图形
- 每个图的每个角表示一个变量，字符串类型会标注在图的下方
- 角线的长度表达值的大小

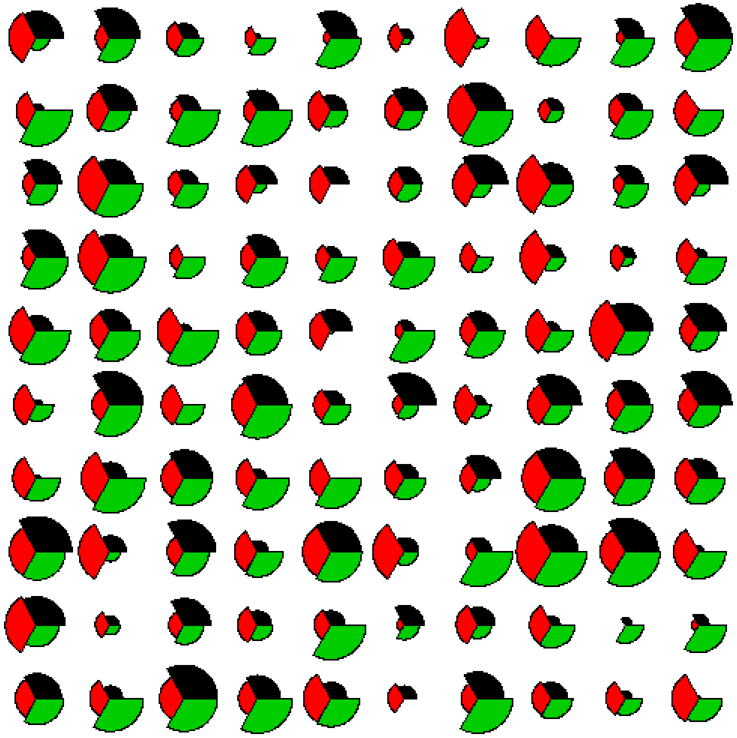


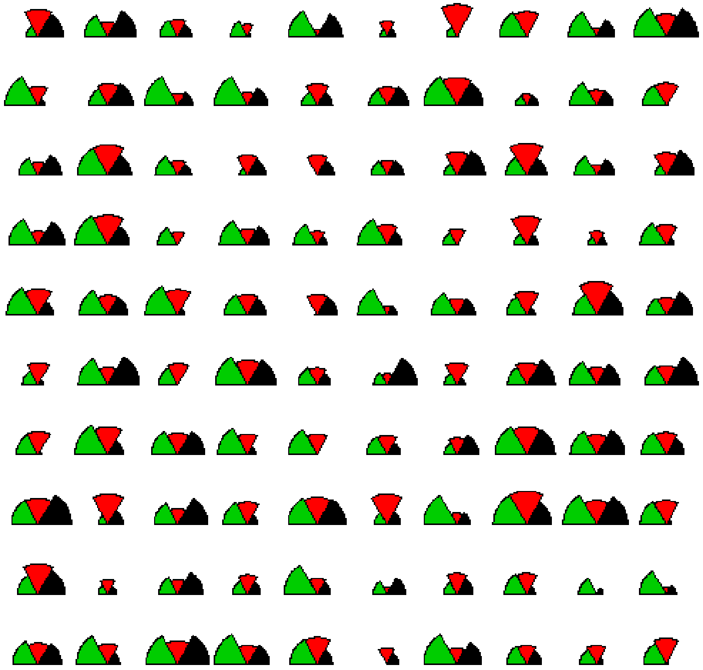
```
> stars(x[c("x1","x2","x3")])
> 
```



星相图

```
> stars(x[c("x1","x2","x3")],full=T,draw.segment=T)
```



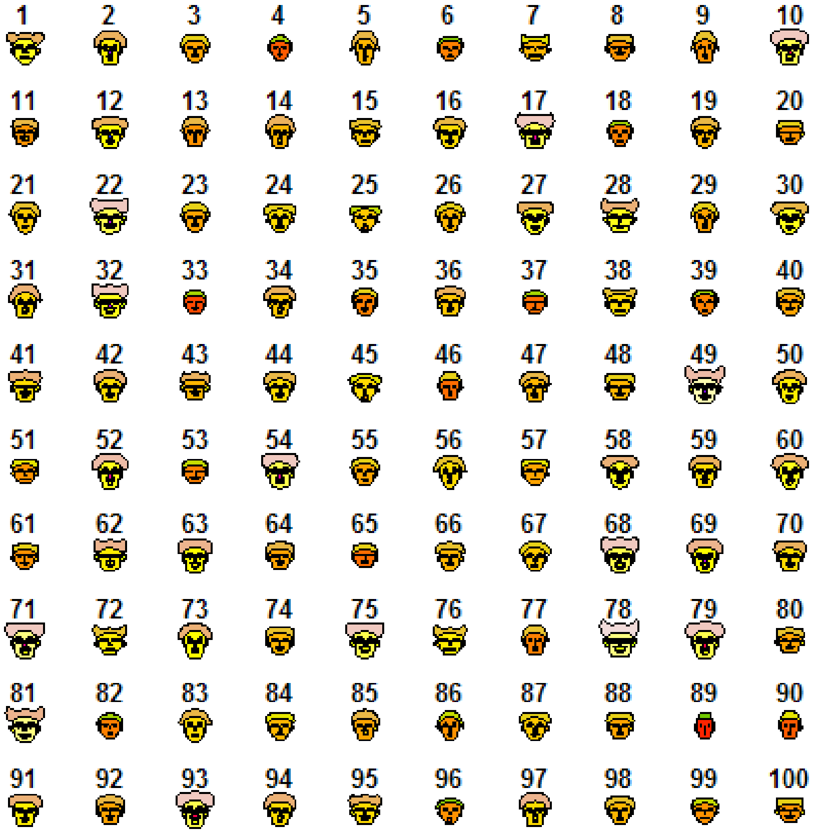


```
> stars(x[c("x1","x2","x3")],full=F,draw.segment=T)
```

脸谱图

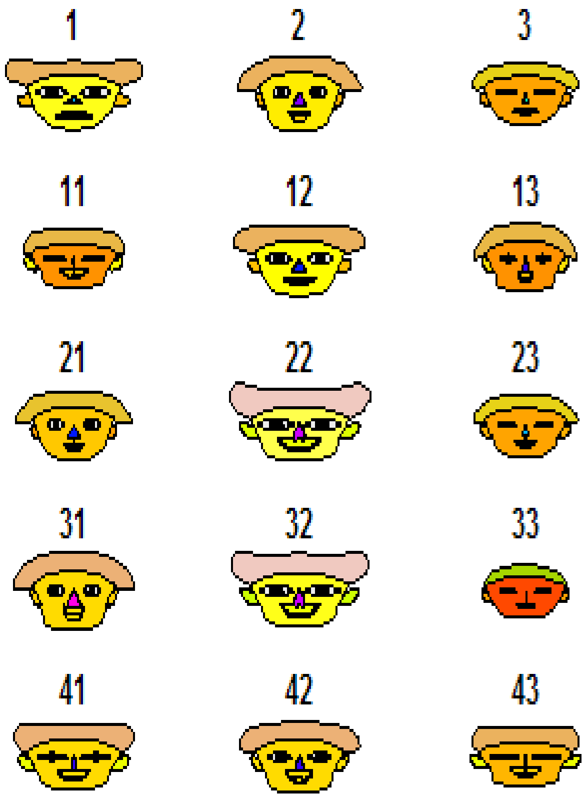
- 安装aplpack包

```
> faces(x[c("x1","x2","x3")])
```



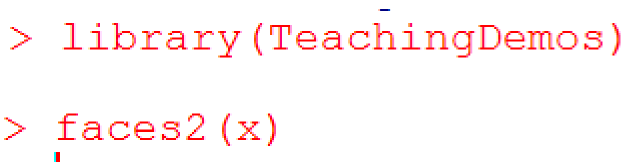
脸谱图

- 用五官的宽度和高度来描绘数值
- 人对脸谱高度敏感和强记忆
- 适合较少样本的情况

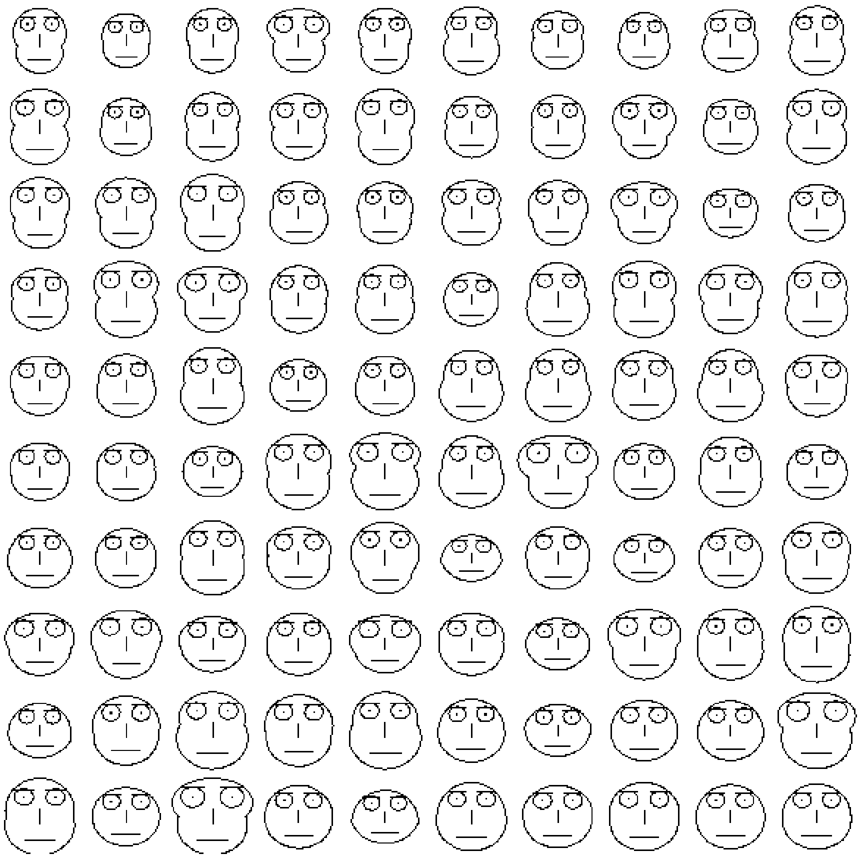


其它脸谱图

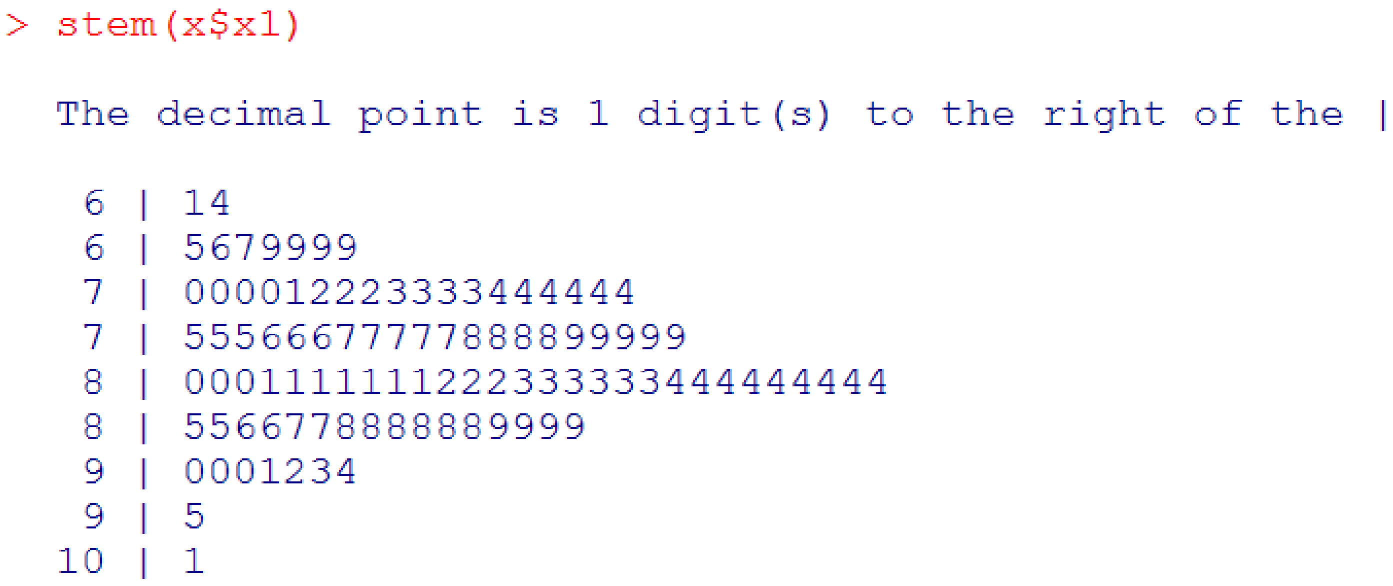
- 安装TeachingDemos包



```
> library(TeachingDemos)
> faces2(x)
```



茎叶图

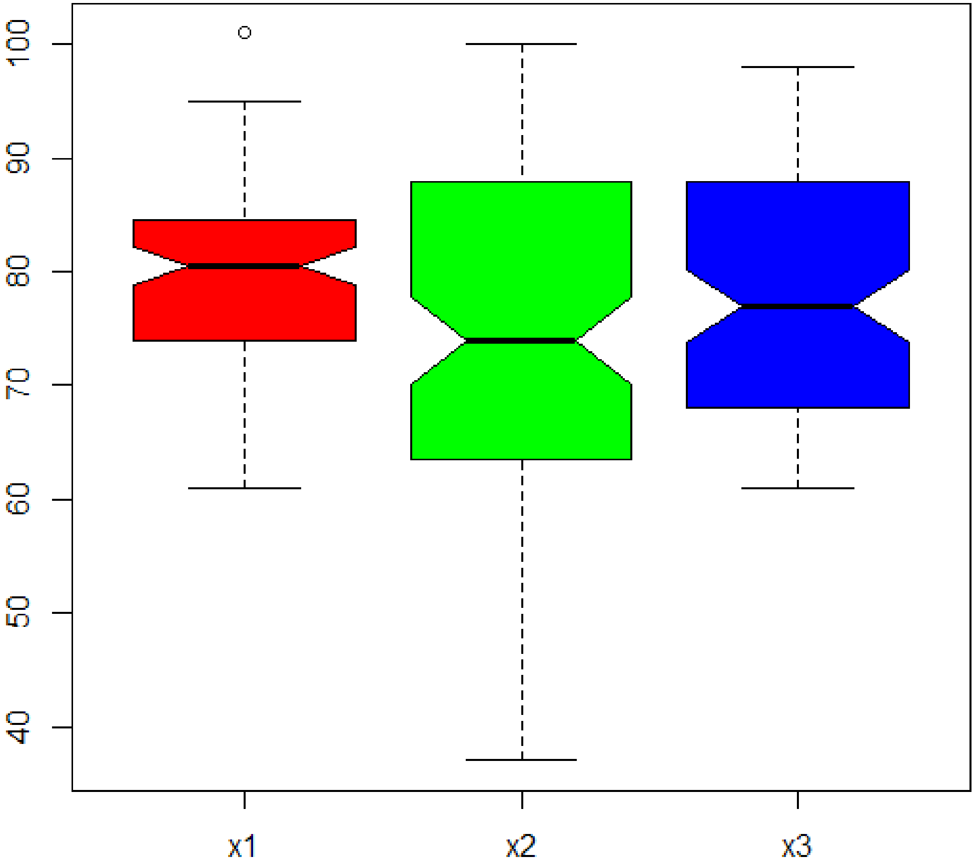


```
> stem(x$x1)

  The decimal point is 1 digit(s) to the right of the |

   6 | 14
   6 | 5679999
   7 | 000012223333444444
   7 | 55566677777888899999
   8 | 0001111111222333333444444444
   8 | 5566778888889999
   9 | 0001234
   9 | 5
  10 | 1
```

2015.3.10

箱线图

```
boxplot(x[2:4],col=c("red","green","blue"),notch=T)
```

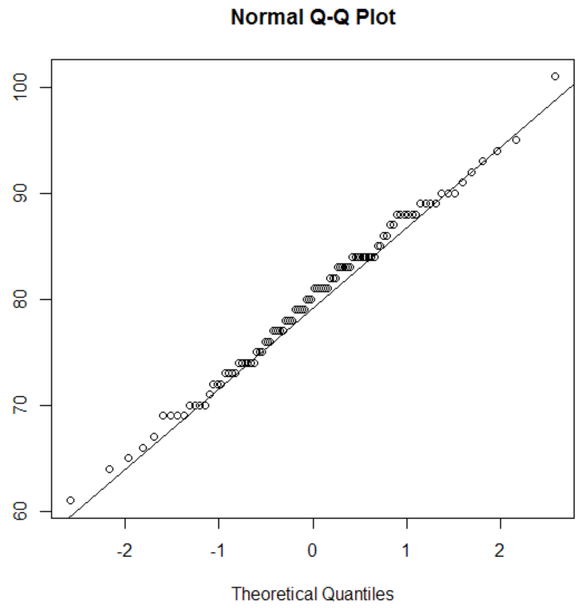


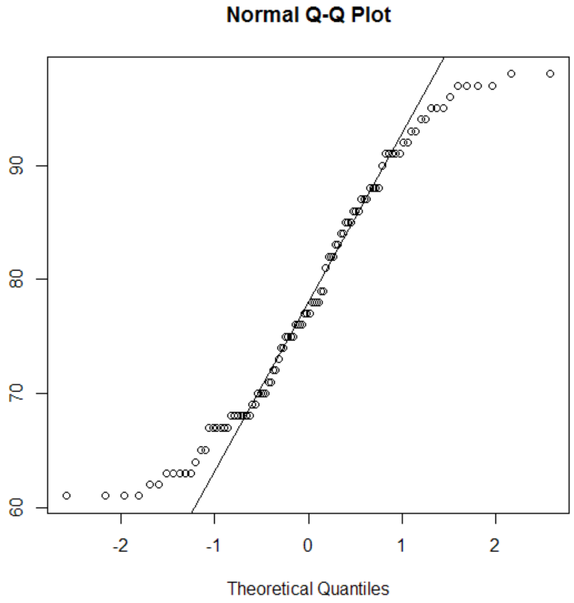
2015.3.10

QQ图

- 可用于判断是否正态分布
- 直线的斜率是标准差，截距是均值
- 点的散布越接近直线，则越接近正态分布

```
> qqnorm(x1)
> qqline(x1)
> qqnorm(x3)
> qqline(x3)
```

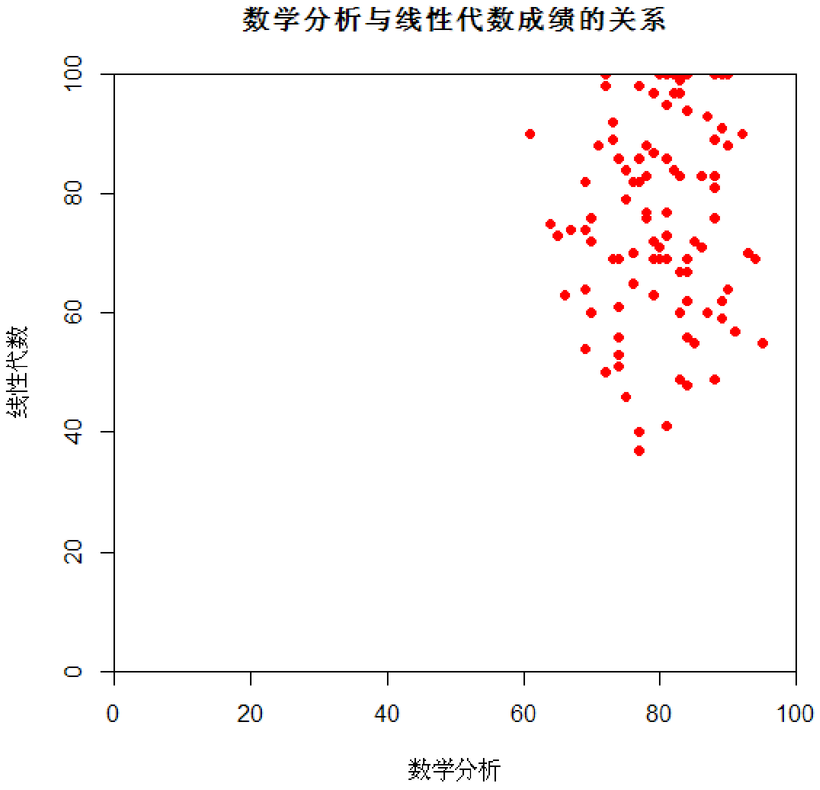




散点图

- 散点图的进一步设置

```
plot(x$x1,x$x2,
main="数学分析与线性代数成绩的关系",
xlab="数学分析",
ylab="线性代数",
xlim=c(0,100),
ylim=c(0,100),
xaxs="i", #Set x axis style as internal
yaxs="i", #Set y axis style as internal
col="red", #Set the color of plotting symbol to red
pch=19) #Set the plotting symbol to filled dots
```

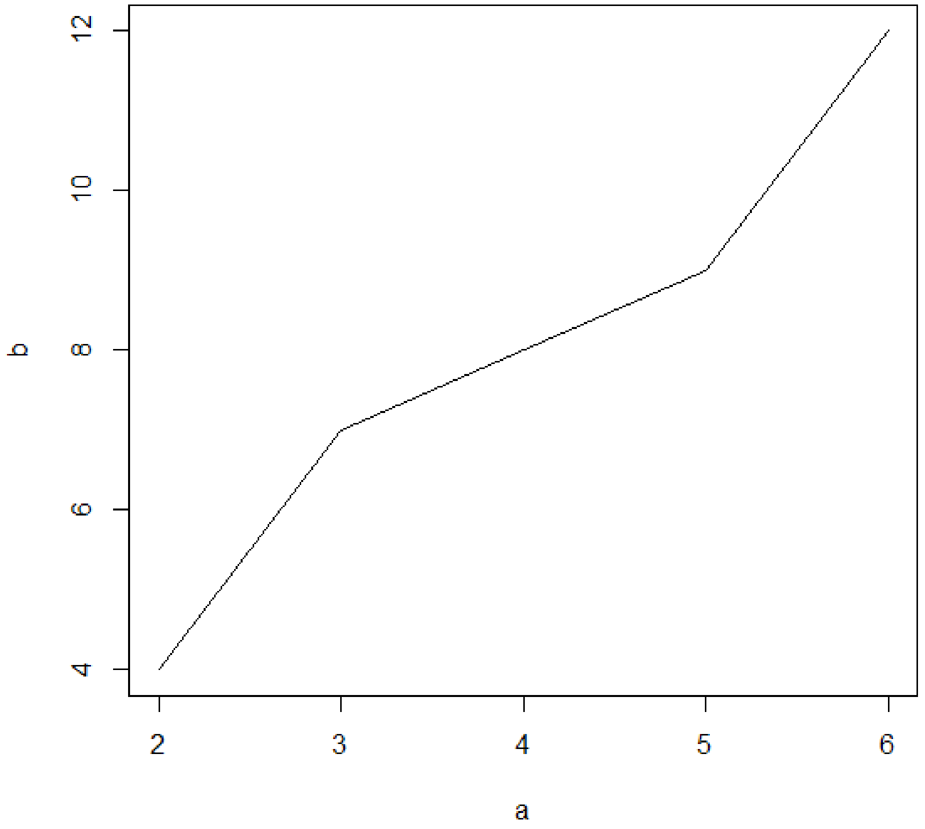


2015.3.10

散点图

- 连线图

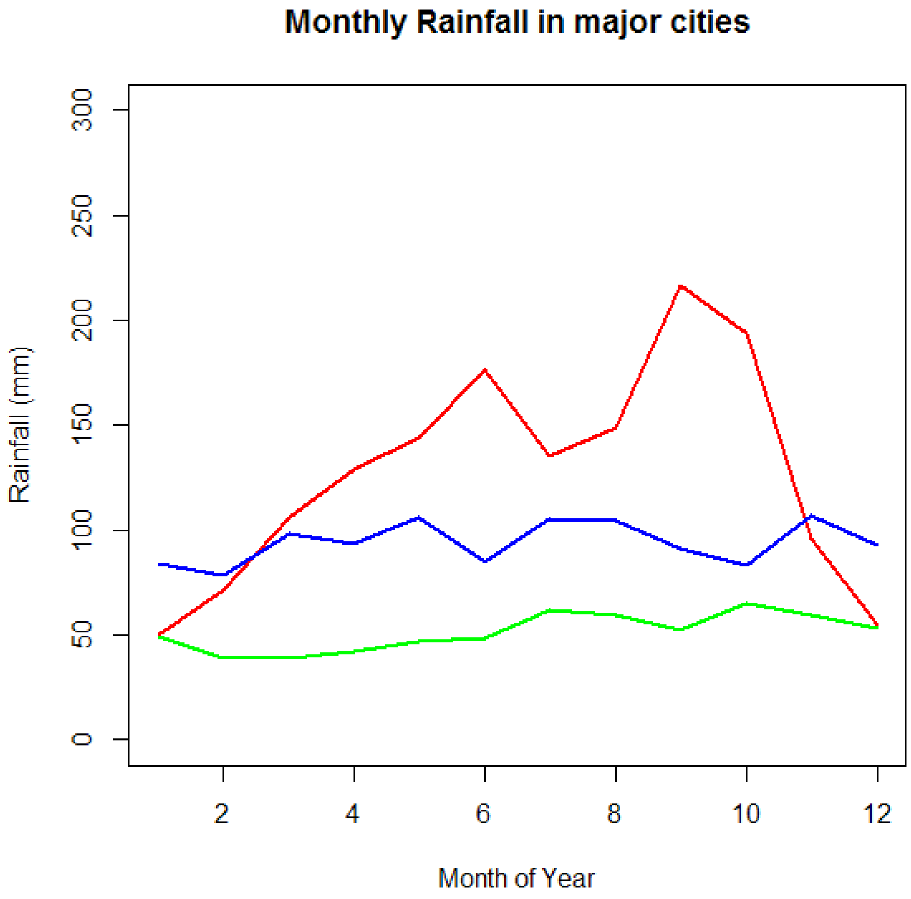
```
a=c(2,3,4,5,6)

b=c(4,7,8,9,12)

plot(a,b,type="l")
```



散点图

- 多条曲线的效果

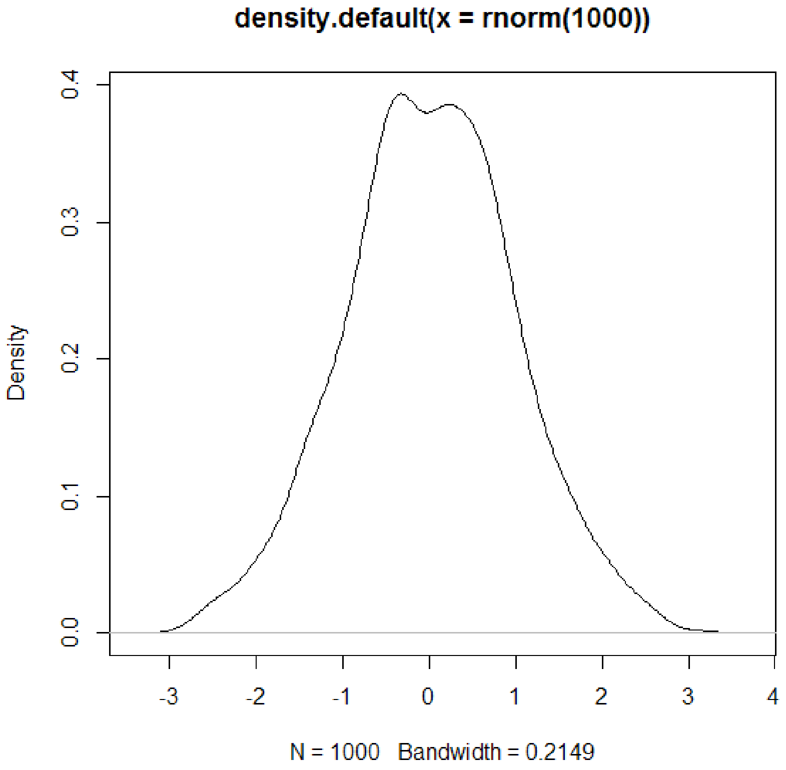
```
plot(rain$Tokyo,type="l",col="red",
ylim=c(0,300),
main="Monthly Rainfall in major cities",
xlab="Month of Year",
ylab="Rainfall (mm)",
lwd=2)
lines(rain$NewYork,type="l",col="blue",lwd=2)
lines(rain$London,type="l",col="green",lwd=2)
lines(rain$Berlin,type="l",col="orange",lwd=2)
```



密度图

- 函数density( )

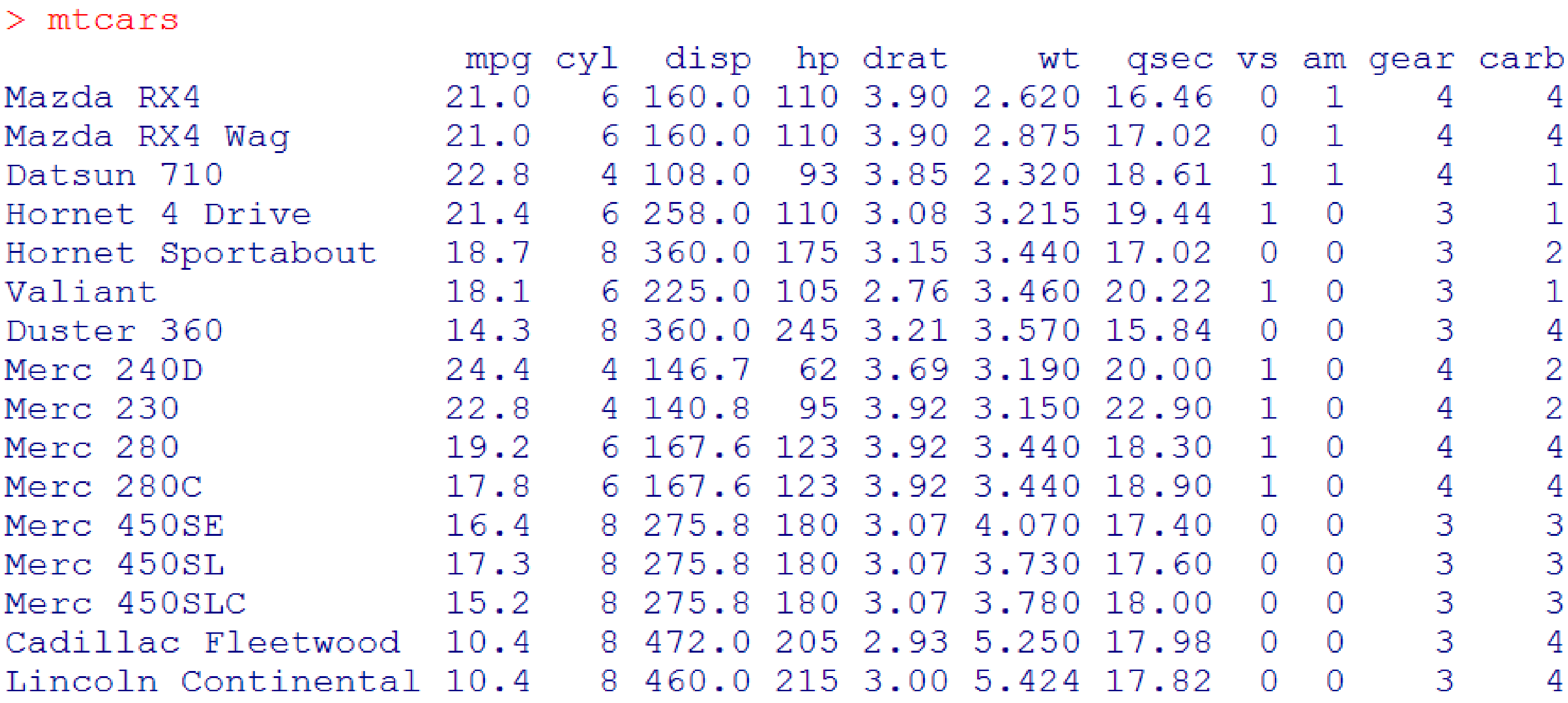
plot(density(rnorm(1000)))



2015.3.10

R内置数据集

- 函数data( )列出内置数据

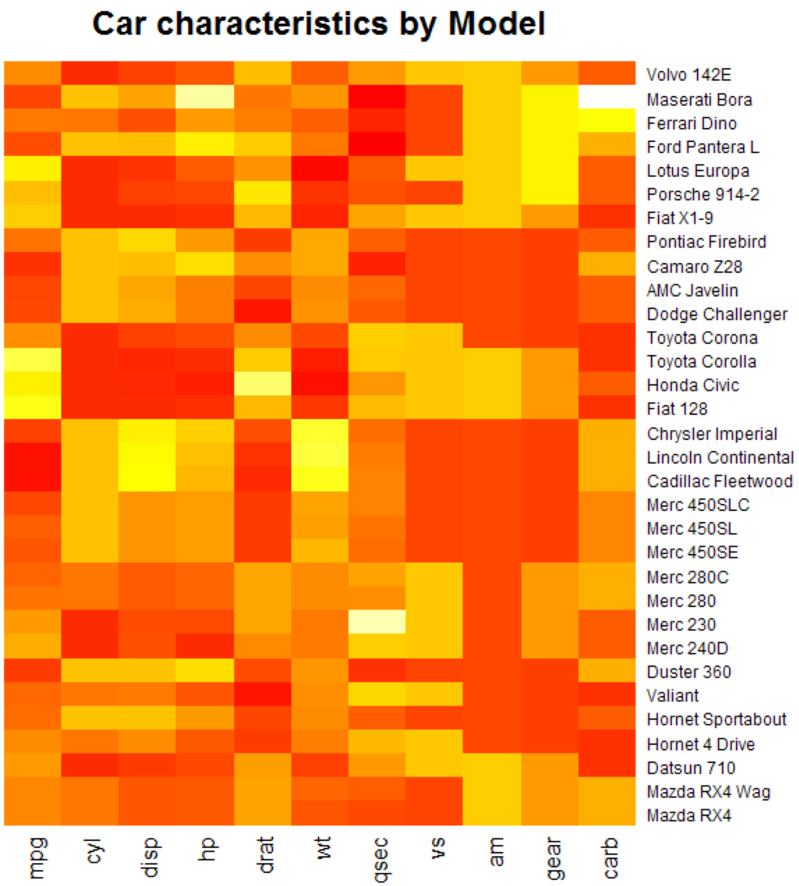


```
> mtcars
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
```

热力图

- 利用内置的mtcars数据集绘制

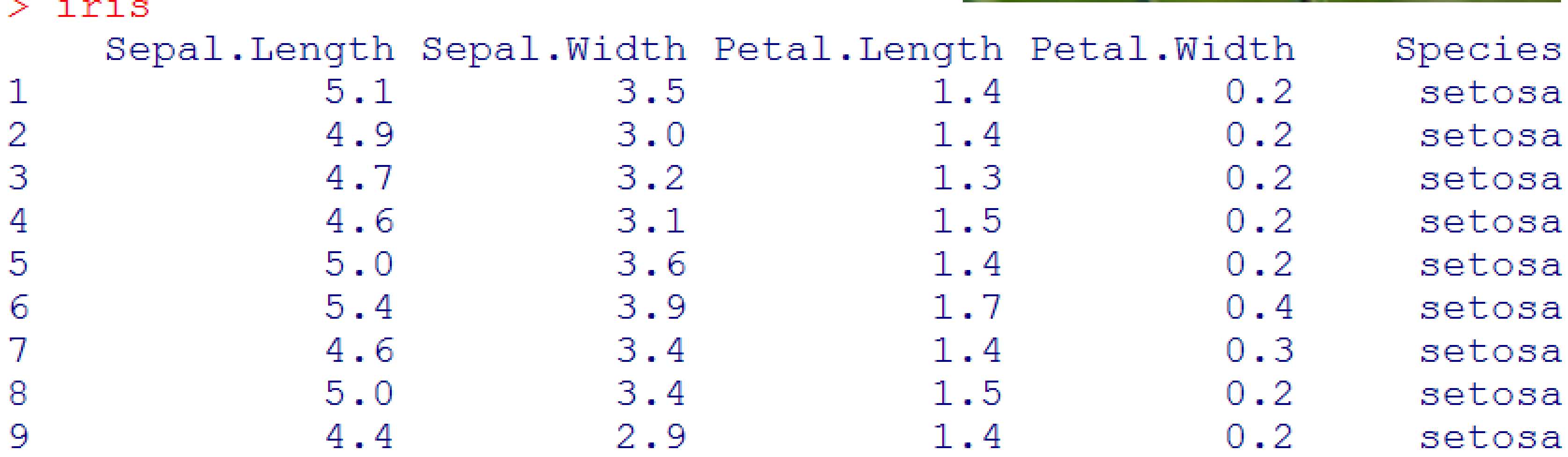
```
heatmap(as.matrix(mtcars),
Rowv=NA,
Colv=NA,
col = heat.colors(256),
scale="column",
margins=c(2,8),
main = "Car characteristics by
    Model")
```



2015.3.10

Iris（鸢尾花）数据集

- Sepal 花萼
- Petal 花瓣
- Species 种属

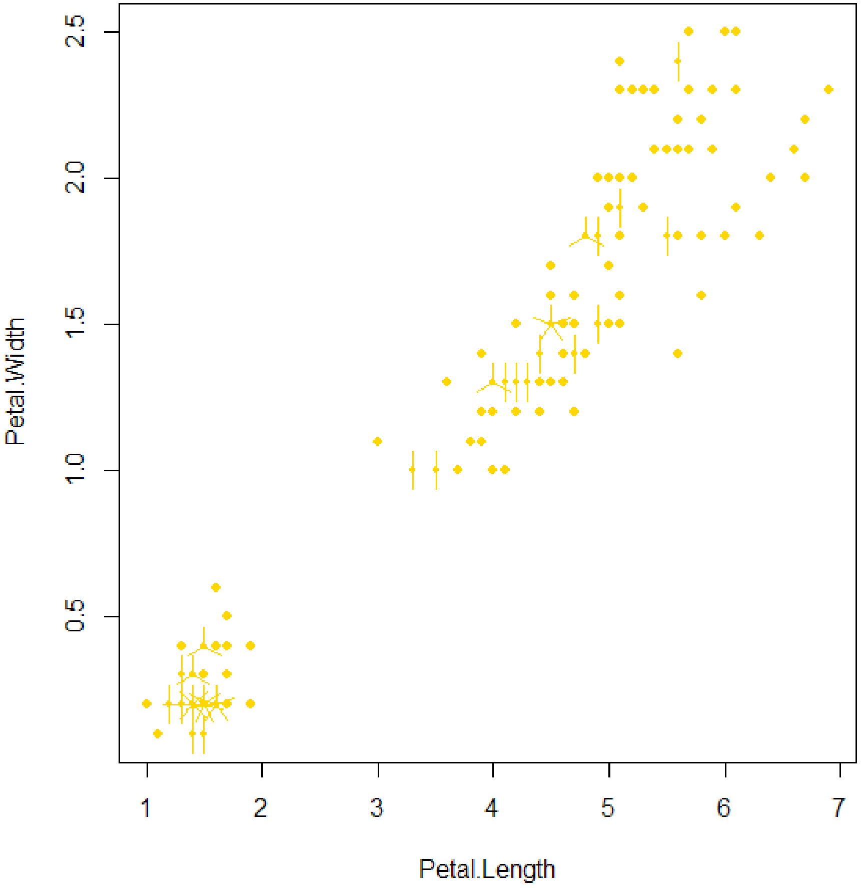


```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
1            5.1         3.5          1.4         0.2    setosa
2            4.9         3.0          1.4         0.2    setosa
3            4.7         3.2          1.3         0.2    setosa
4            4.6         3.1          1.5         0.2    setosa
5            5.0         3.6          1.4         0.2    setosa
6            5.4         3.9          1.7         0.4    setosa
7            4.6         3.4          1.4         0.3    setosa
8            5.0         3.4          1.5         0.2    setosa
9            4.4         2.9          1.4         0.2    setosa
```

向日葵散点图

- 用来克服散点图中数据点重叠问题
- 在有重叠的地方用一朵“向日葵花”的花瓣数目来表示重叠数据的个数

```
sunflowerplot(iris[, 3:4], col =
    "gold", seg.col = "gold")
```

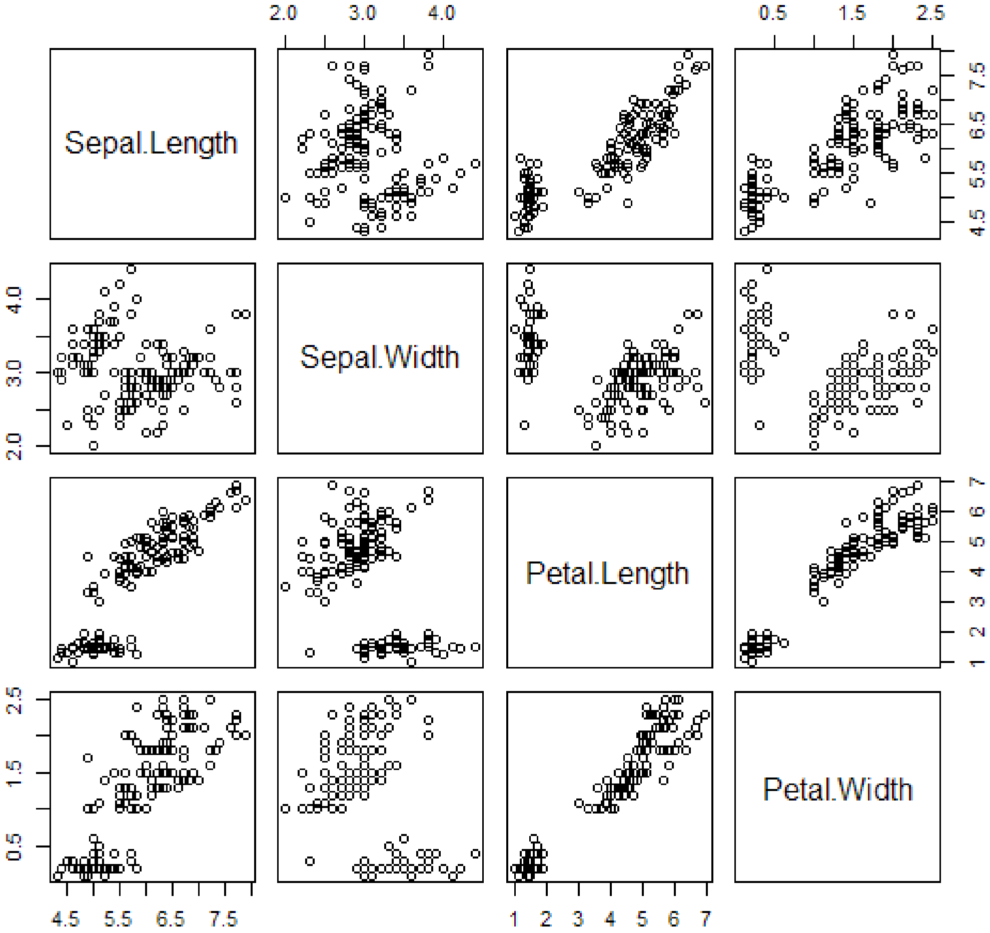


2015.3.10

散点图集

- 遍历样本中全部的变量配对画出二元图
- 直观地了解所有变量之间的关系

pairs(iris[,1:4])

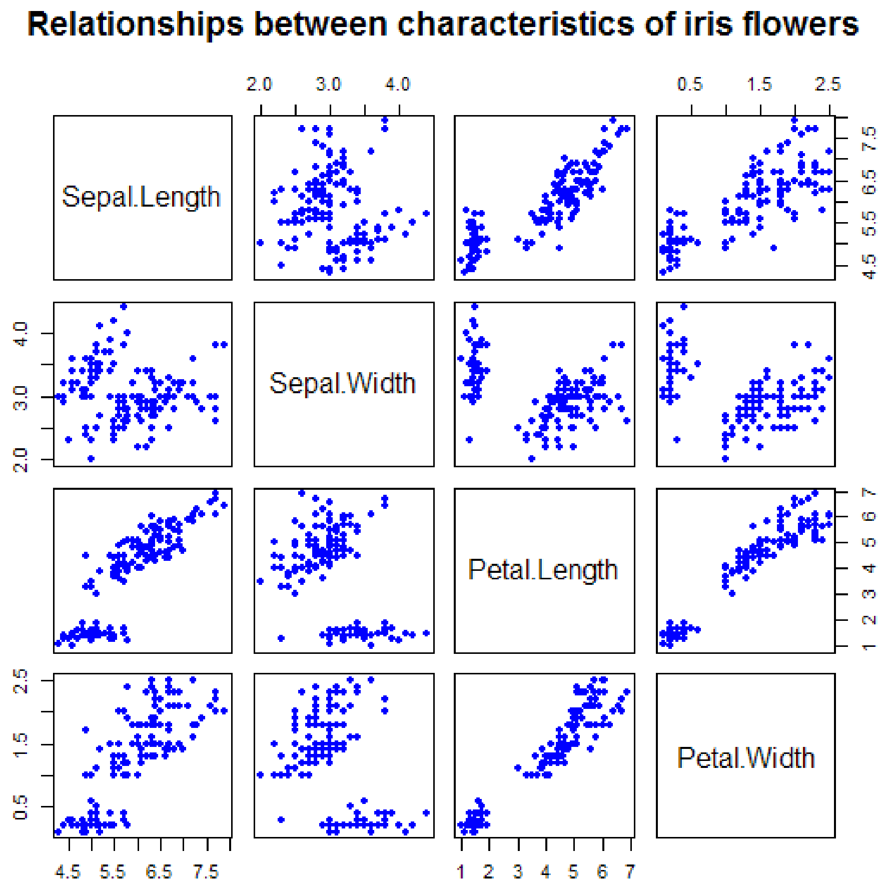


2015.3.10

散点图集

- 用plot也可以实现同样的效果

```
plot(iris[,1:4],
main="Relationships between
      characteristics of iris flowers",
pch=19,
col="blue",
cex=0.9)
```

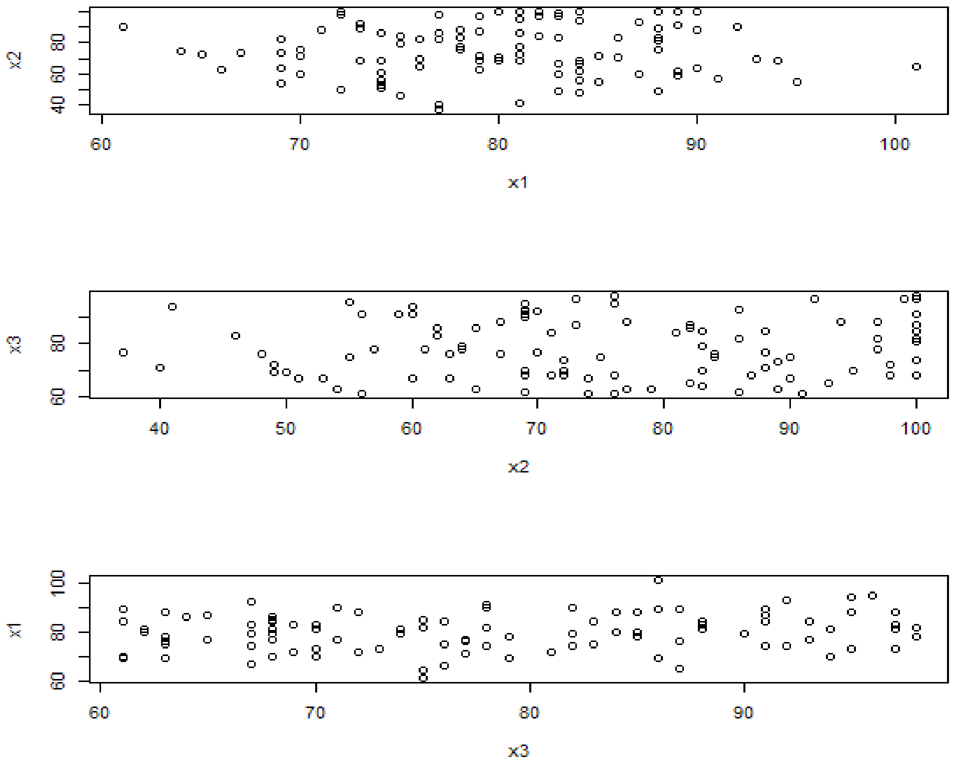


2015.3.10

散点图集

- 利用par( )在同一个device输出多个散点图
- Par命令博大精深，用于设置绘图参数，help(par)

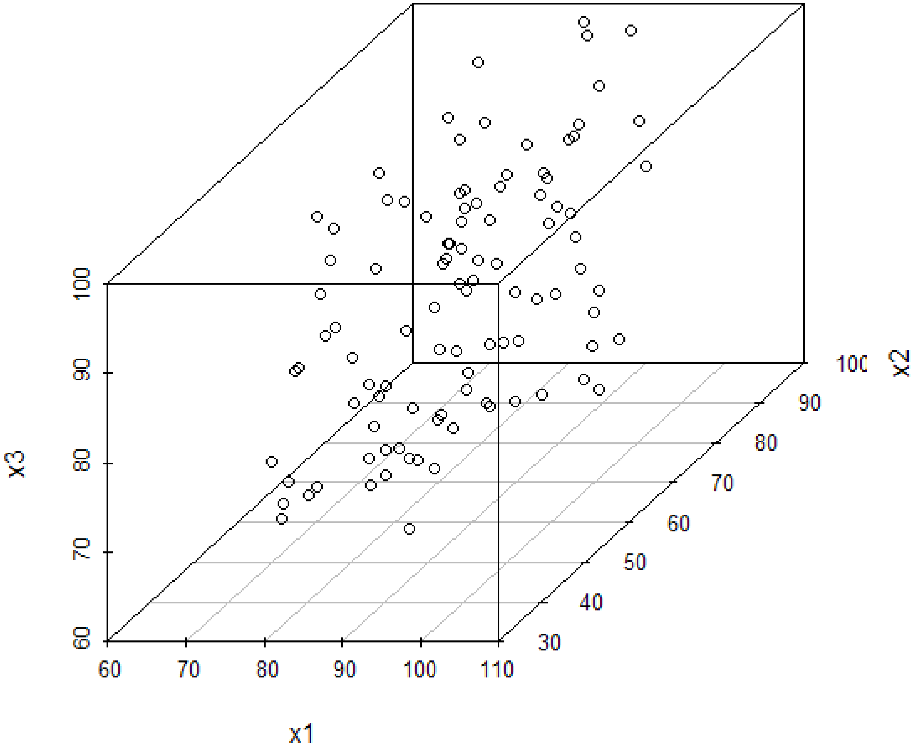
```
par(mfrow=c(3,1))
plot(x1,x2);plot(x2,x3);plot(x3,x1)
```



三维散点图

- 安装scatterplot3d 包

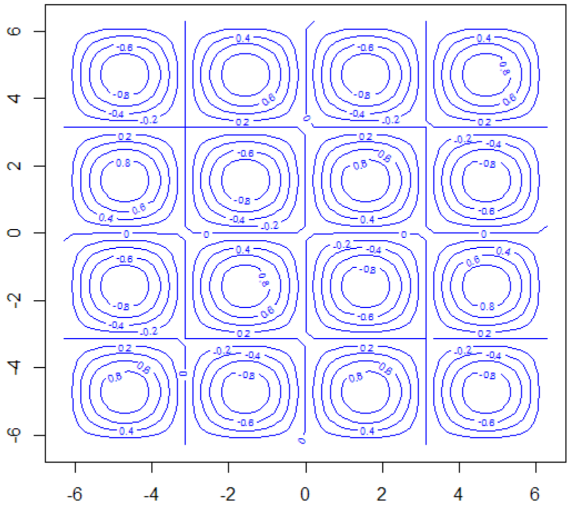
scatterplot3d(x[2:4])

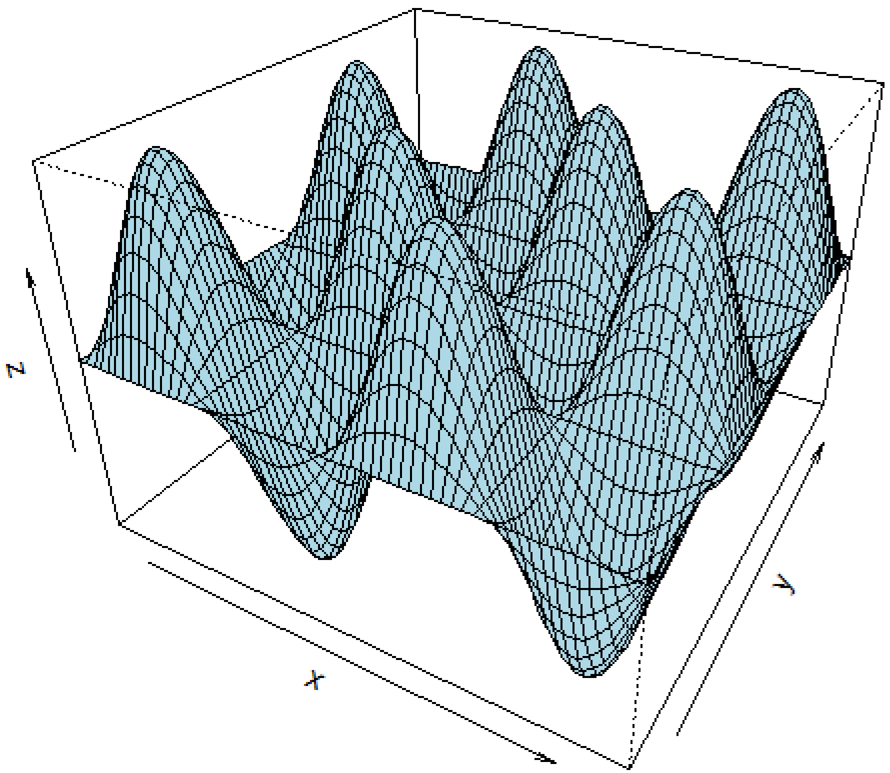


2015.3.10

三维作图

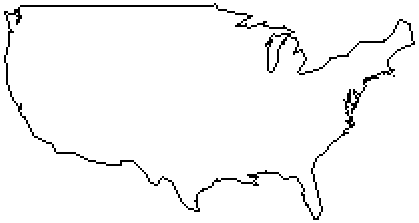
```
x<-y<-seq(-2*pi, 2*pi, pi/15)
f<-function(x,y) sin(x)*sin(y)
z<-outer(x, y, f)
contour(x,y,z,col="blue")
persp(x,y,z,theta=30, phi=30,
      expand=0.7,col="lightblue")
```



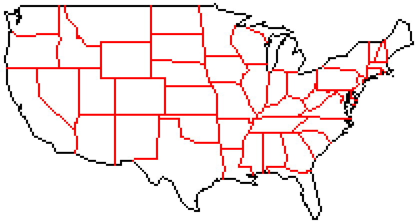


地图

- 安装maps包

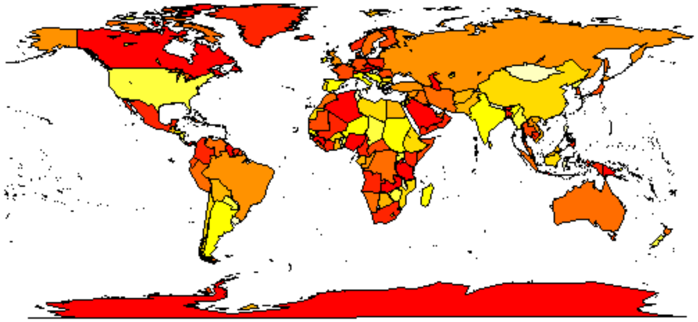


map("state", interior = FALSE)

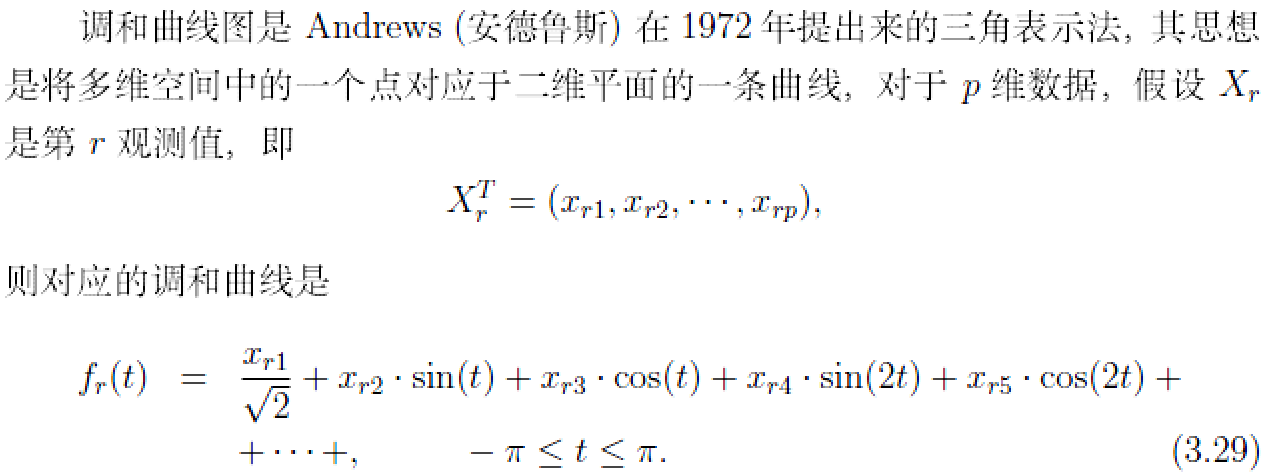


map("state", boundary = FALSE, col="red",
    add = TRUE)

map('world', fill = TRUE,col=heat.colors(10))



调和曲线图



调和曲线图是 Andrews (安德鲁斯) 在 1972 年提出来的三角表示法，其思想是将多维空间中的一个点对应于二维平面的一条曲线，对于 p 维数据，假设 X_r 是第 r 观测值，即

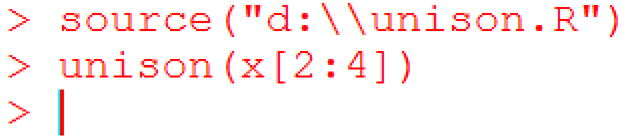
$$X_r^T = (x_{r1}, x_{r2}, \cdots, x_{rp}),$$

则对应的调和曲线是

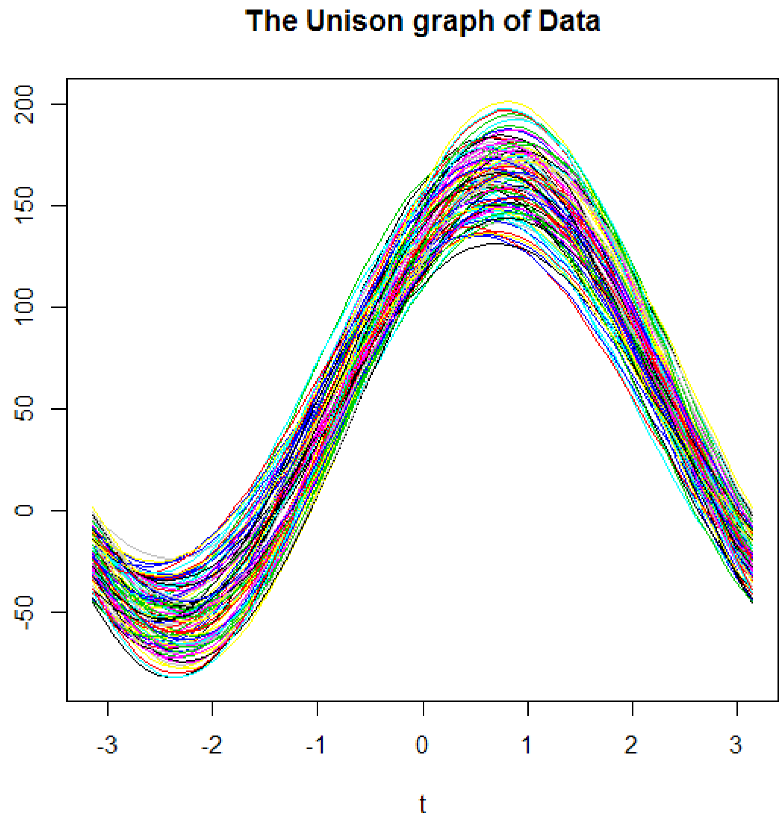
$$\begin{aligned} f_r(t) &= \frac{x_{r1}}{\sqrt{2}} + x_{r2} \cdot \sin(t) + x_{r3} \cdot \cos(t) + x_{r4} \cdot \sin(2t) + x_{r5} \cdot \cos(2t) + \\ &\quad + \cdots +, \qquad -\pi \le t \le \pi. \end{aligned} \tag{3.29}$$

调和曲线图

- unison.r的代码
- 自定义函数
- 调和曲线用于聚类判断非常方便



```
> source("d:\\unison.R")
> unison(x[2:4])
> 
```



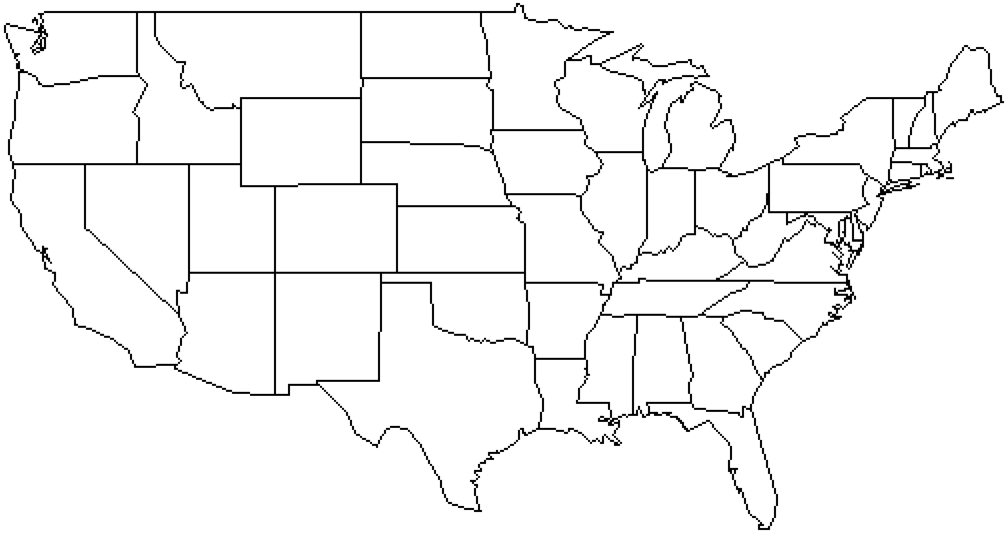
R实验：社交数据可视化

- 先下载安装maps包和geosphere包并加载

```
library(maps)
library(geosphere)
```

- 画出美国地图

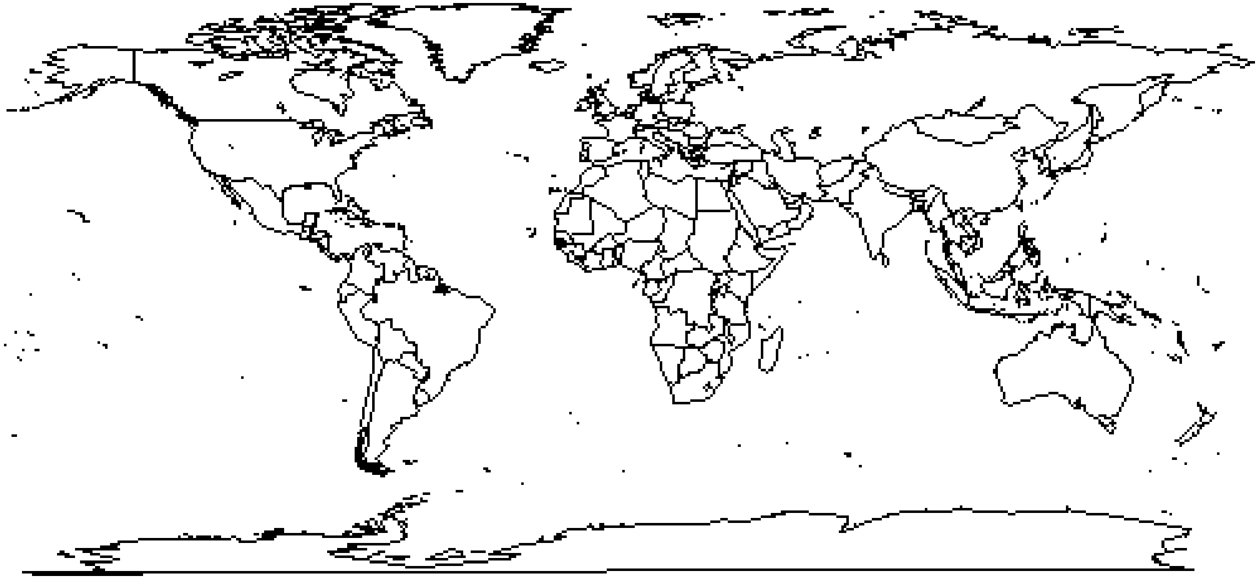
```
map("state")
```



R实验：社交数据可视化

- 画世界地图

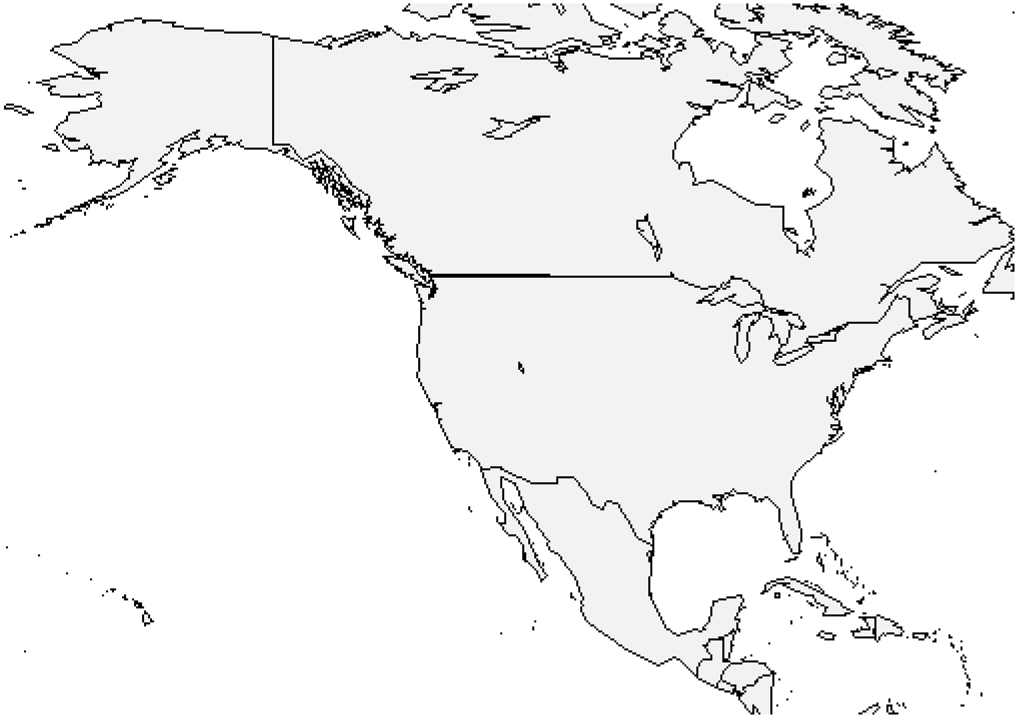
```
map("world")
```



R实验：社交数据可视化

- 通过设置坐标范围使焦点集中在美国周边，并且设置一些有关颜色

```
xlim <- c(-171.738281, -56.601563)
ylim <- c(12.039321, 71.856229)
map("world", col="#f2f2f2", fill=TRUE, bg="white", lwd=0.05, xlim=xlim, ylim=ylim)
```

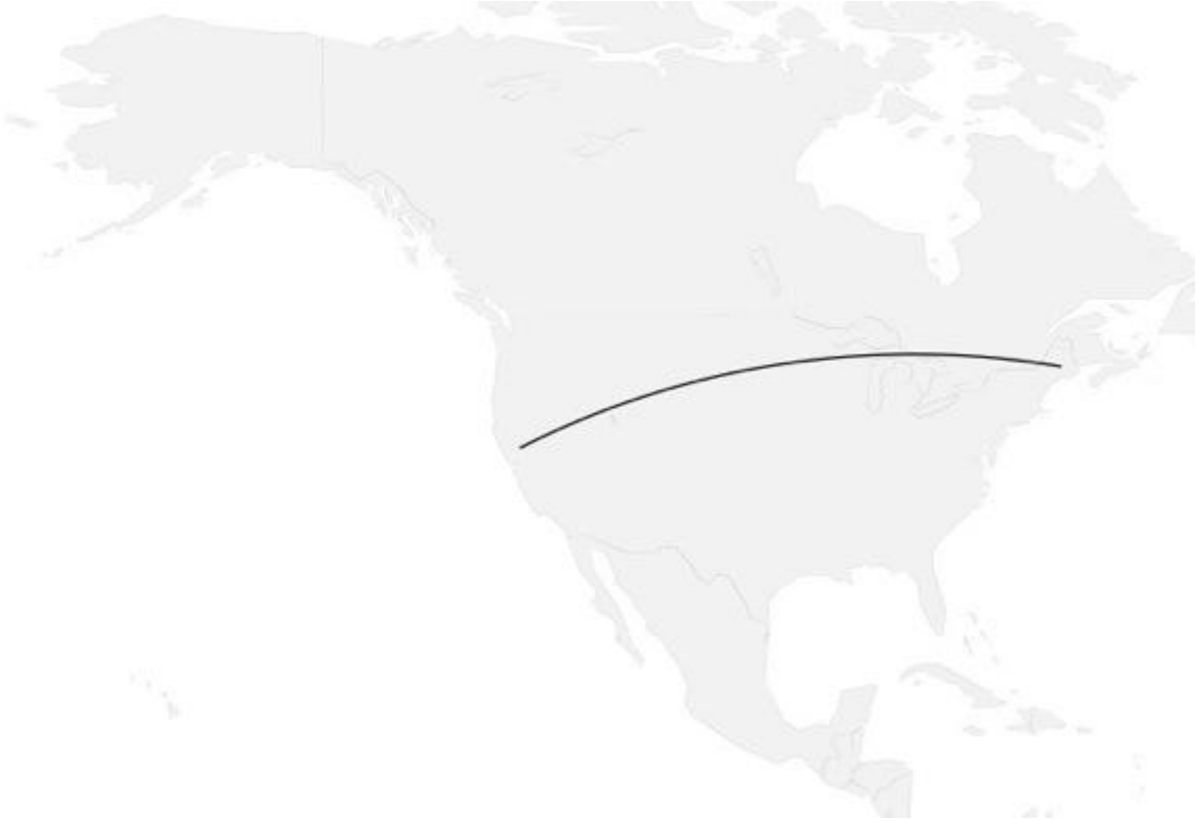


2015.3.10

R实验：社交数据可视化

- 画一条弧线连线，表示社交关系

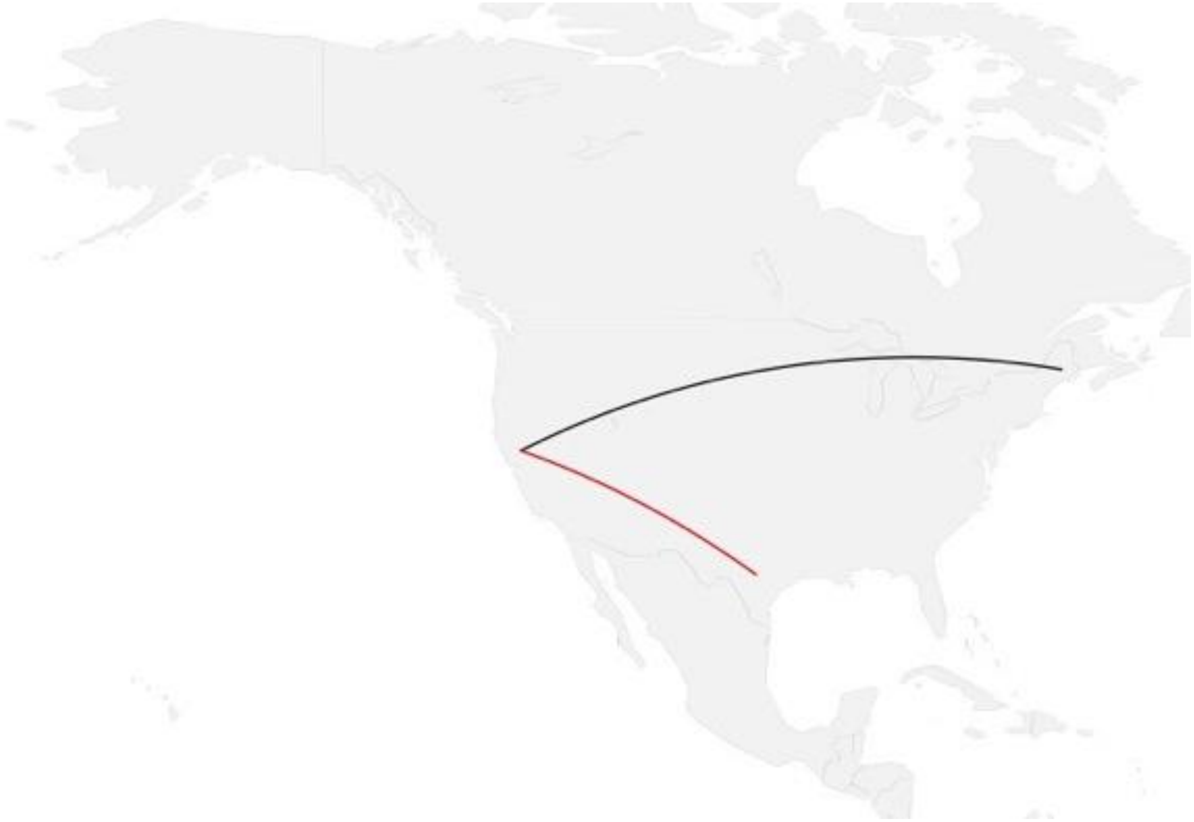
```
lat_ca <- 39.164141
lon_ca <- -121.64062
5lat_me <- 45.21300
4lon_me <- -68.906250
inter <-
    gcIntermediate(c(lon_c
    a, lat_ca), c(lon_me,
    lat_me), n=50,
    addStartEnd=TRUE)
lines(inter)
```



R实验：社交数据可视化

- 继续画弧线

```
lat_tx <- 29.954935
lon_tx <- -98.701172
inter2 <-
    gcIntermediate(c(lon_ca
    , lat_ca), c(lon_tx, lat_tx),
    n=50,
    addStartEnd=TRUE)
lines(inter2, col="red")
```



R实验：社交数据可视化

- 装载数据

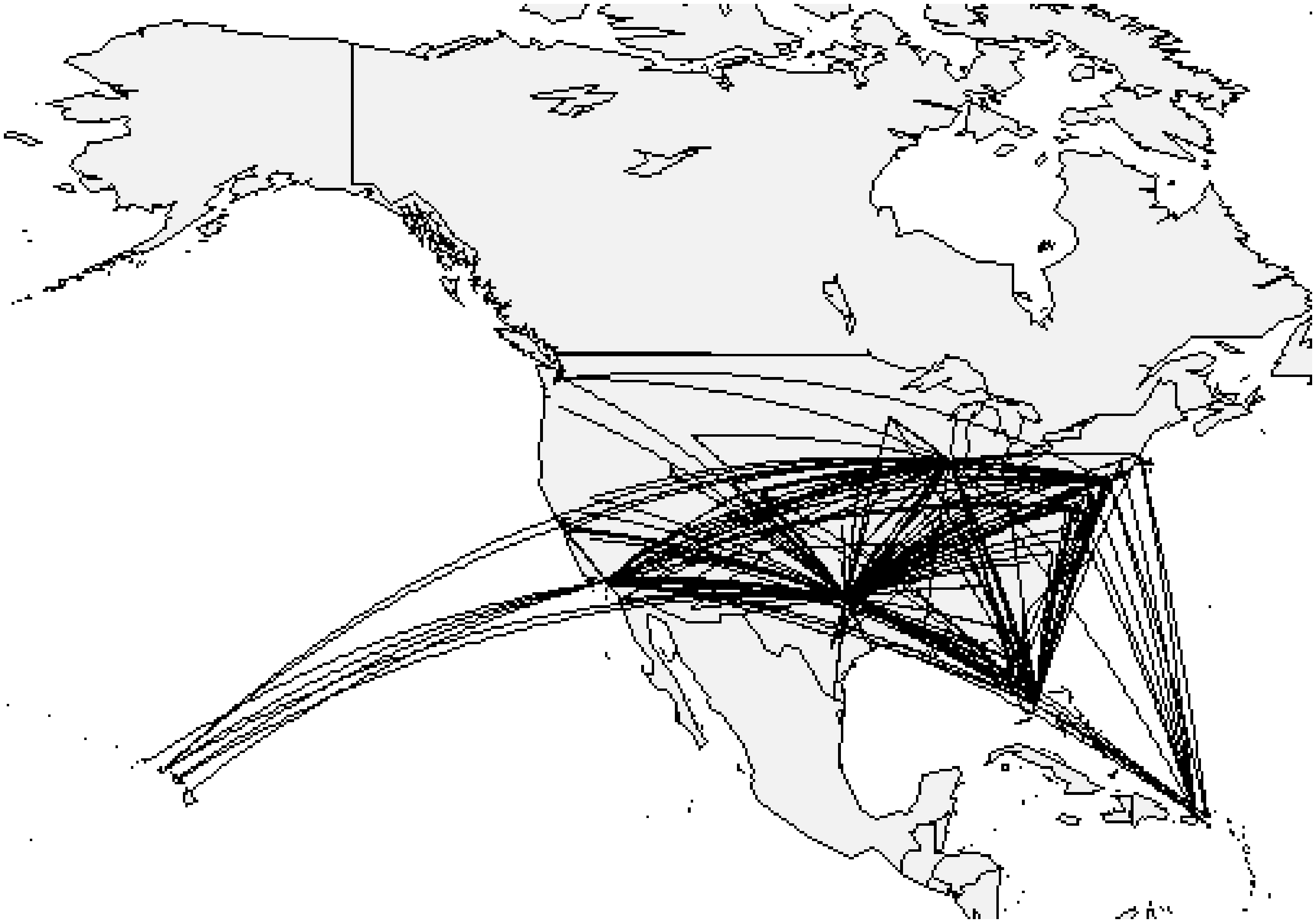
```
airports <- read.csv("http://datasets.flowingdata.com/tuts/maparcs/airports.csv",
    header=TRUE)
flights <- read.csv("http://datasets.flowingdata.com/tuts/maparcs/flights.csv",
    header=TRUE, as.is=TRUE)
```

R实验：社交数据可视化

- 画出多重联系

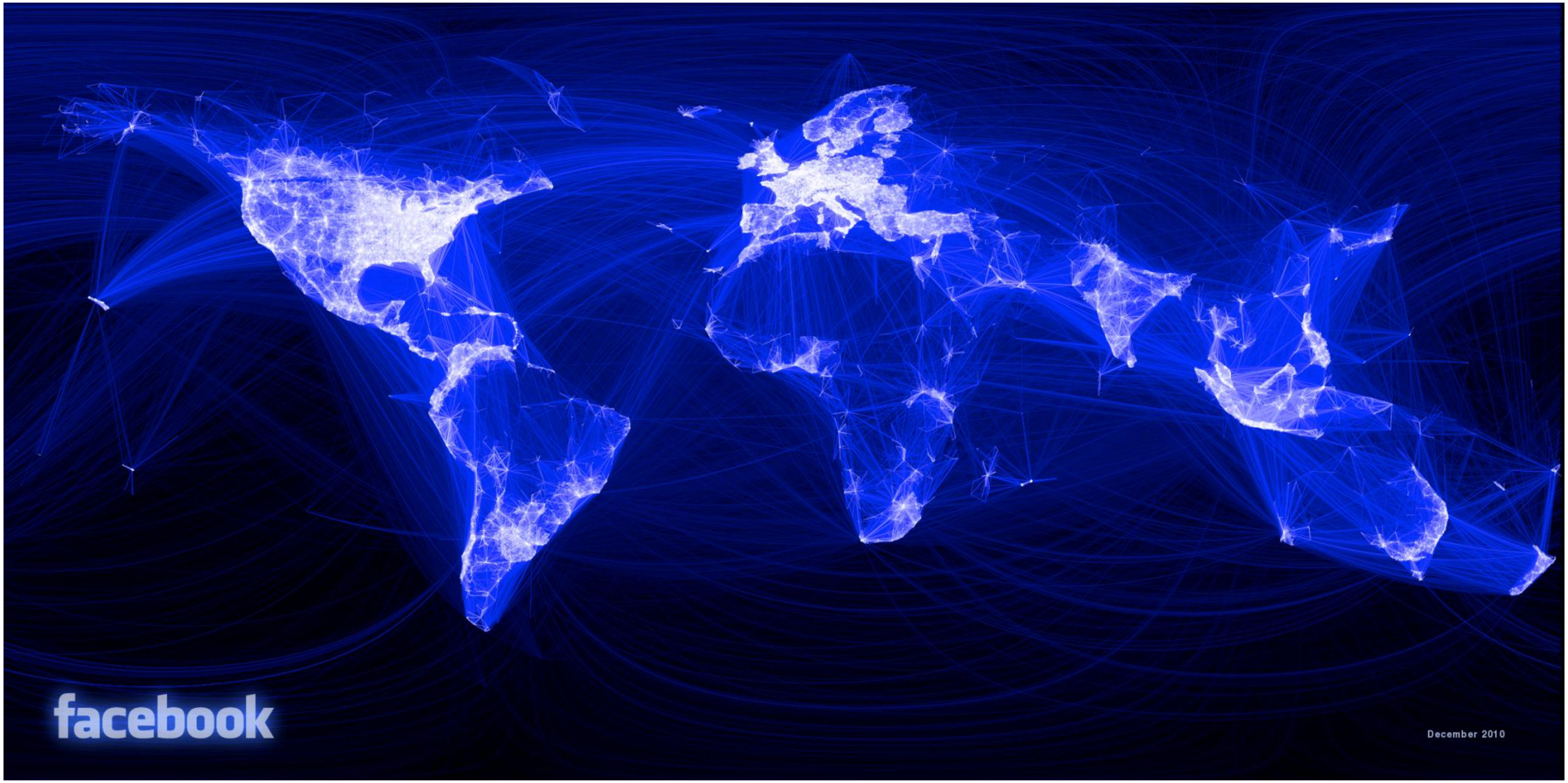
```
map("world", col="#f2f2f2", fill=TRUE, bg="white", lwd=0.05, xlim=xlim, ylim=ylim)


fsub <- flights[flights$airline == "AA",]
for (j in 1:length(fsub$airline)) {
    air1 <- airports[airports$iata == fsub[j,]$airport1,]
    air2 <- airports[airports$iata == fsub[j,]$airport2,]

    inter <- gcIntermediate(c(air1[1,]$long, air1[1,]$lat), c(air2[1,]$long, air2[1,]$lat), n=100,
    addStartEnd=TRUE)

    lines(inter, col="black", lwd=0.8)
}
```

R实验：社交数据可视化



2015.3.10

R实验：社交数据可视化



http://flowingdata.com/2011/05/11/how-to-map-connections-with-great-circles/

Thanks

FAQ时间