R 语言入门

1.1 R 语言概述

网络舆情的研究中经常需要通过一些分析工具以实现对舆情热点的分析挖掘，在数据爆发的当下，大数据时代的来临带来了新的产物，基于网络舆情的背景下，通过大数据的分析挖掘方法对舆情热点进行分析，对于研究内容的知识发现与规则提取，具有相当的现实意义。而进行大数据分析时，通常需要各种软件和程序语言来实现，在各种统计分析软件中简单、方便、易于操作的便是 R。

那么，什么是 R?

R 语言是由奥克兰大学的 Ross Ihaka 和 Robert Gentleman 开发的面向对象语言，它是一个免费开源，可以在各个操作系统上安装使用的统计分析软件，提供了各种统计分析算法和数据展现方法，R 语言的前身是 S 语言，可认为是 S 语言的一种实现。

为什么使用 R?

大多数人使用 R 的主要原因在于其简单方便，界面友好，拥有非常多的第三方程序包，很适合初学者学习。IEEE 发布 2016 年度编程语言排行榜中，R 语言位居第五名，足见 R 的应用之广。由于 R 的开源，任何人都可以为 R 社区提供高效的源代码或程序包，这些程序包审核通过后可直接被使用者下载安装使用，从而扩大了 R 的功能，在网络舆情分析中，常用的算法如：聚类，关联规则，决策树，文本挖掘，情感分析等，在 R 中均有相应程序包的功能实现，通过下载，安装，加载等步骤即可使用，操作简单。

图 1-1　IEEE 发布的 2016 年编程语言排行榜前十名

R 的运行环境很多，用户可以直接安装 R，也可以通过安装集成开发环境，R 的常见集成开发环境有 RStudio，Rgui 等，本章节，将会介绍 R 和 RStudio 的安装及工作目录。

1.2 R 的下载安装与使用

1.2.1 安装 R

R 作为免费开源的统计分析软件，用户可通过 R 官网直接下载，官方网站为：https://www.r-project.org，进入官网，将会看到图 1-2 所示，在 R 官网中，可以下载 R 的软件或程序包（在左侧的 CRAN 中进行下载），也可以浏览 R 官方信息。

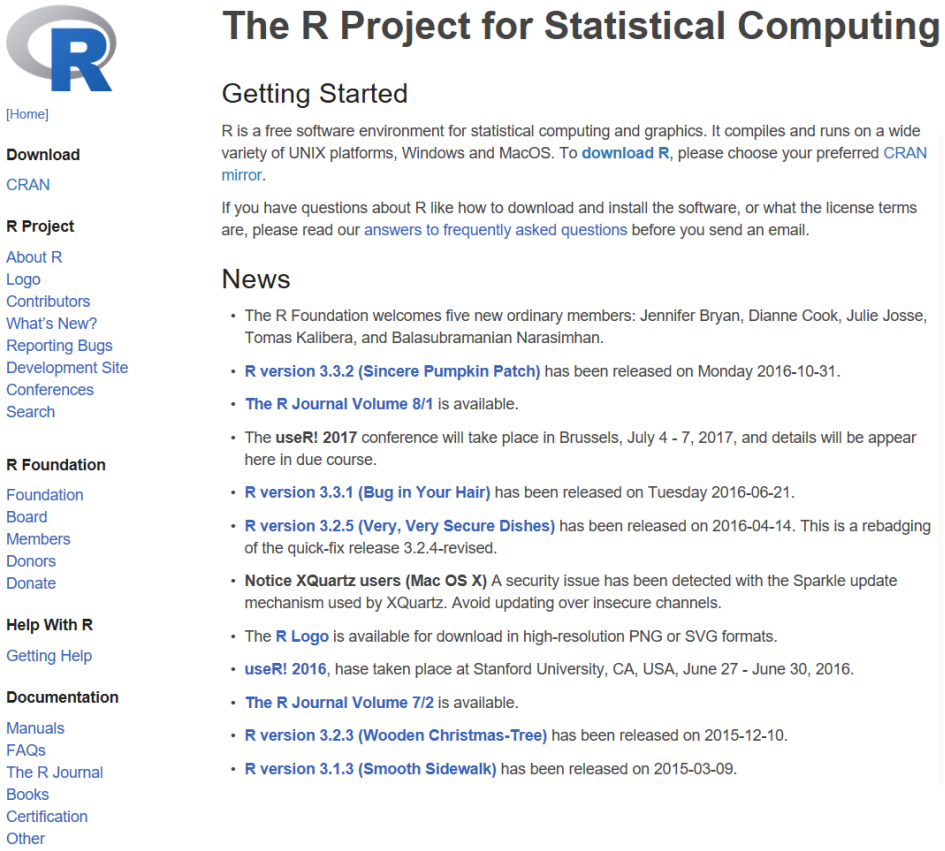


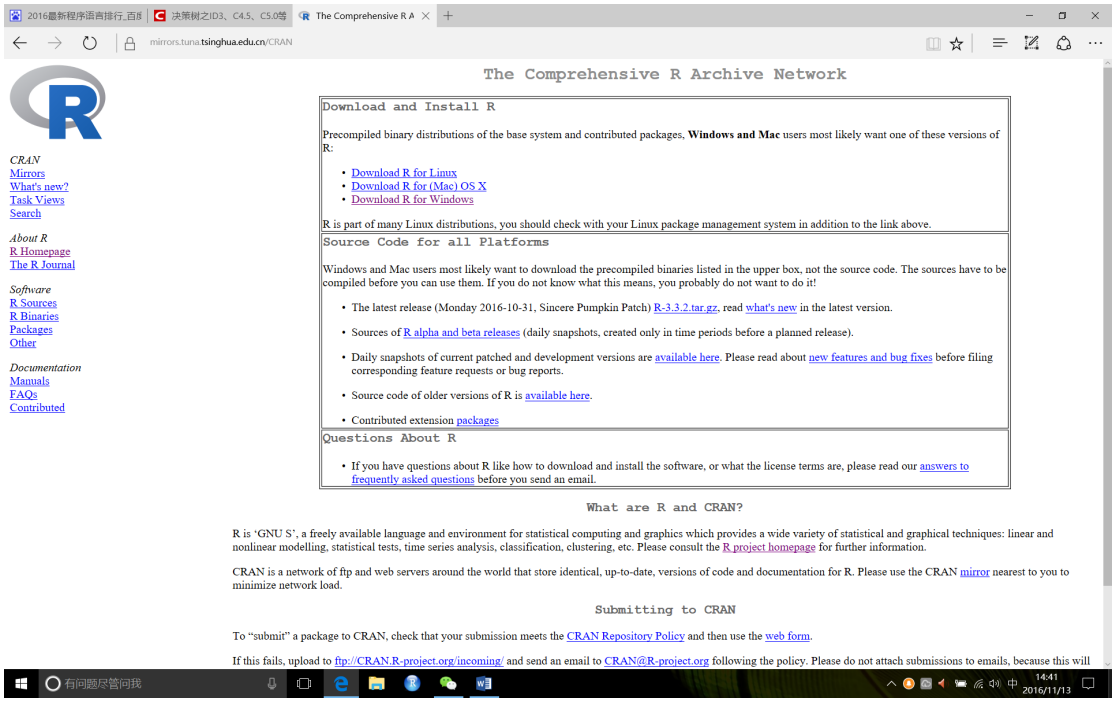
图 1-2　R 官网首页

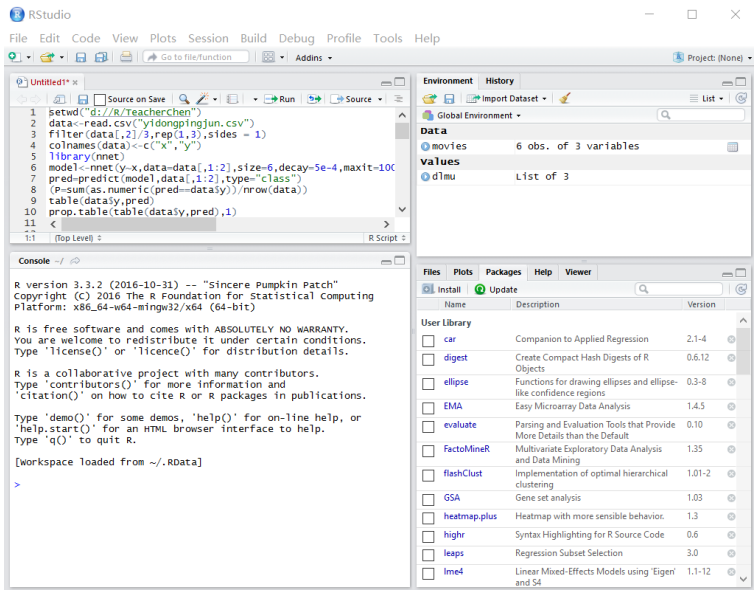
点击左侧 CRAN 后，选择 China 中的镜像，以 http://mirrors.tuna.tsinghua.edu.cn/CRAN/ 为例进行操作。

China

https://mirrors.tuna.tsinghua.edu.cn/CRAN/
http://mirrors.tuna.tsinghua.edu.cn/CRAN/
https://mirrors.ustc.edu.cn/CRAN/
http://mirrors.ustc.edu.cn/CRAN/
https://mirror.lzu.edu.cn/CRAN/
http://mirror.lzu.edu.cn/CRAN/
http://mirrors.xmu.edu.cn/CRAN/

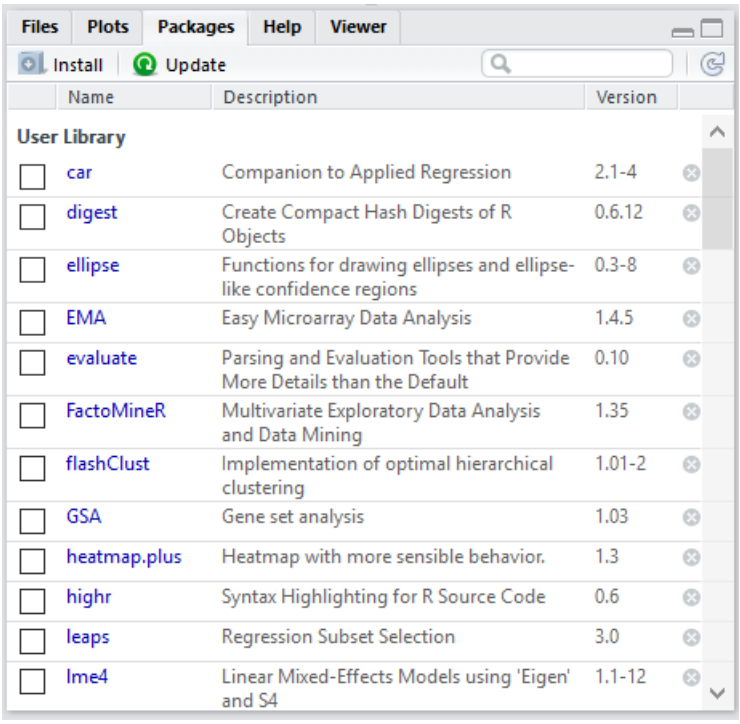
进入下载网页，可选择操作系统进行下载



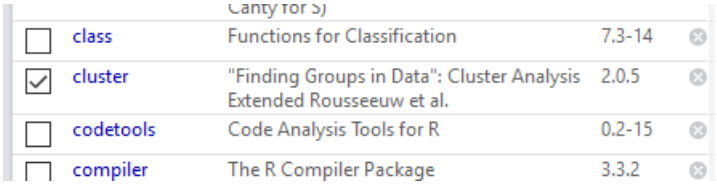


1.2.2 R 的扩展包

可以这么理解，R 是函数与扩展程序包的集合，在 R 的操作环境下，我们可以通过安装、加载扩展包来实现对数据的处理。安装程序包可执行语句：install.packages(程序包名)实现，在 Rstudio 中，右下方的工作空间内设有 install 按钮，单击可进行程序包安装。



程序包安装完成后，还不能使用，需要对程序包进行加载，通常使用 library()函数或 require()函数，在 Rstudio 中，也可以在右下方工作空间内找到相应的程序包，在前面勾选后，即可使用。



1.3 R 的基本介绍

1.3.1 数学运算、变量和字符串

在 R 中进行基本的运算操作简单方便，如简单四则运算及乘方、开方操作如下：

```
4+2  #加
4-2  #减
4*2  #乘
4/2  #除
4^2  #乘方
sqrt(4)  #开方
```

在 R 中，为变量赋值可使用<-符号（也可以使用“=”进行赋值，但不常用），如下代码将字符串“Hello World”赋值给变量 a

```
> a <- "Hello World"
> a
[1] "Hello World"
```

1.3.2 向量及其运算

向量是一维对象数组，在 R 中对向量对象的类型没有限制，上面代码中已经接触，可通过 c（）命令创建向量，对于向量创建的操作，还有以下常用的便捷方法：

创建 1 到 10 之间的连续自然数：

```
> c(1:10)
 [1]  1  2  3  4  5  6  7  8  9 10
```

创建 1 到 10 之间的奇数集合：

```
> c((1:5)*2-1)
[1] 1 3 5 7 9
```

向量的类型可以使数值向量、字符串向量、逻辑向量等，逻辑向量只含有真（True）和假（False）两种，可通过比较获得：

```
> a<-c(1:10)
> a>5
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
>
> a<-c(1:10)
> a
 [1]  1  2  3  4  5  6  7  8  9 10
> a > 5
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

由于 R 的统计分析特征，R 中的函数可以进行向量计算，例如：median()函数用于求中位数，sum()函数用于求和，max()/min()函数用于求最大/小值，mean()函数用于求平均数，sd()函数用于求方差等，其操作如下：

```
a<-c(1,5,3,6,8,9,5,3,0,4)  #将数列赋值给变量a
median(a)   #求a的中位数
sum(a)   #求a中各数的和
max(a)   #求a中的最大值
min(a)   #求a中的最小值
mean(a)   #求a的各数的平均值
sd(a)   #求a中各数的方差
```

执行结果如下：

```
> a<-c(1,5,3,6,8,9,5,3,0,4)  #将数列赋值给变量a
> median(a)   #求a的中位数
[1] 4.5
> sum(a)   #求a中各数的和
[1] 44
> max(a)   #求a中的最大值
[1] 9
> min(a)   #求a中的最小值
[1] 0
> mean(a)   #求a的各数的平均值
[1] 4.4
> sd(a)   #求a中各数的方差
[1] 2.836273
```

如果想得到向量的更多统计量，可对向量应用 summary（）函数，如下图，可得到向量 a 的最小/大值、平均数、中位数、第一/三四分位数等信息。

```
> a <- c(1,5,3,6,8,9,5,3,0,4)
> summary(a)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    3.00    4.50    4.40    5.75    9.00
```

如果想得到向量中第 2 个元素的值，可进行如下操作：

```
> a[2]
[1] 5
```

也可以在[]中写入向量，下面代码表示从向量 a 中选取第 2 个和第三个元素：

```
> a[c(2,3)]
[1] 5 3
```

想删除向量中的元素，可在该元素索引前加“-”：

```
> a[-2]
[1] 1 3 6 8 9 5 3 0 4
```

在 R 中，也可以对向量进行四则运算操作，如下：

向量与数的四则运算：

```
> c(5:8)+2
[1]  7  8  9 10
> c(5:8)-2
[1] 3 4 5 6
> c(5:8)*2
[1] 10 12 14 16
> c(5:8)/2
[1] 2.5 3.0 3.5 4.0
```

向量与向量的四则运算：

```
> c(5:8)+c(1:4)
[1]  6  8 10 12
> c(5:8)-c(1:4)
[1] 4 4 4 4
> c(5:8)*c(1:4)
[1]  5 12 21 32
> c(5:8)/c(1:4)
[1] 5.000000 3.000000 2.333333 2.000000
```

通过运算结果可以发现，向量与向量的四则运算只是将对应索引的元素值进行计算，不同索引值之间无影响。此外，如果两个不同长度的向量进行运算时，长度短的向量会被循环使用，如下图：

```
> c(5:8)+c(1,2)
[1]  6  8  8 10
> c(5:8)-c(1,2)
[1] 4 4 6 6
> c(5:8)*c(1,2)
[1]  5 12  7 16
> c(5:8)/c(1,2)
[1] 5 3 7 4
```

1.4 R 中的数据结构

1.4.1 数组与矩阵

数组是含有同类型数据的结构，可以多维，存在多行多列，想要创建数组，需先创建向量，再指明行列数，如下：

```
> v <- c(1:10)
> v_array <- array(v,dim = c(3,4))
> v_array
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8    1
[3,]    3    6    9    2
```

需要说明的是，制定的数组为三行四列，即 12 个元素，而向量中只有 10 个元素，因此，从第 11 个数起开始循环，这与向量间运算类似。以下介绍数组的基本操作：

```
> v_array[12]
[1] 2
>
> v_array[1,2]
[1] 4
>
> v_array[1,2] #访问数组第一行第二列的元素
[1] 4
> v_array[1,] #访问数组第一行的元素
[1]  1  4  7 10
> v_array[,2] #访问数组第二列的元素
[1] 4 5 6
> v_array[,-2] #访问数组中除第二列以外的其他元素
     [,1] [,2] [,3]
[1,]    1    7   10
[2,]    2    8    1
[3,]    3    9    2
```

矩阵的结构与数组类似，但矩阵必须是二维，而数组可以有多维，创建过程与数组类似，如下图所示：

```
> v <- c(1:10)
> v_matrix<-matrix(v,nrow = 2,ncol = 5)
> v_matrix
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

选取矩阵的一个子集：

```
> v_matrix[1:2,2:3]
     [,1] [,2]
[1,]    3    5
[2,]    4    6
```

矩阵和数组都是按列进行优先存储的，如果想以行有限存储，则需在创建时设置属性 byrow=TRUE，如下图：

```
> v <- c(1:10)
> v_matrix<-matrix(v,nrow = 2,ncol = 5,byrow = TRUE)
> v_matrix
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
```

1.4.2 列表

在 R 中，列表是对象的集合，与向量类似，但不同的是，列表中的元素根据数据类型而不同，如下图所示，创建了一个以字符串、数字、向量为元素的列表：

```
> dlmu <- list("dalian maritime university",1909,c("dmu","dlmu"))
> dlmu
[[1]]
[1] "dalian maritime university"

[[2]]
[1] 1909

[[3]]
[1] "dmu"   "dlmu"
```

访问列表中的元素操作简单，只需在[]中打出对应元素的索引值，如下图所示，访问列表中第二个元素：

```
> dlmu[2]
[[1]]
[1] 1909
```

也可以访问列表中的一些元素：

```
> dlmu[2:3]
[[1]]
[1] 1909

[[2]]
[1] "dmu"   "dlmu"
```

为了方便查询与读取，我们也可以为列表中的元素赋名字，这在创建列表时需要声明，同时，在读取元素时用$符号，如下图所示：

```
> dlmu <- list(name="dalian maritime university",year=1909,shortname=c("dmu","dlmu"))
> dlmu
$name
[1] "dalian maritime university"

$year
[1] 1909

$shortname
[1] "dmu"   "dlmu"

> dlmu$name
[1] "dalian maritime university"
```

对列表中元素的增加、修改和删除等操作，可以通过赋值语句实现，赋值元素名称如不在列表中，即为增加元素；在列表中，即为修改；赋 NULL 值，即为删除，如下图：

```
> dlmu["age"] <- 108
> dlmu
$name
[1] "dalian maritime university"

$year
[1] 1909

$shortname
[1] "dmu"   "dlmu"

$age
[1] 108

> dlmu["name"] <- "Dalian Maritime University"
> dlmu
$name
[1] "Dalian Maritime University"

$year
[1] 1909

$shortname
[1] "dmu"   "dlmu"

$age
[1] 108

> dlmu["age"] <- NULL
> dlmu
$name
[1] "Dalian Maritime University"

$year
[1] 1909

$shortname
[1] "dmu"   "dlmu"
```

1.4.3 数据框

数据框是一种包含相关信息的数据结构，比如下图数据框包含了多个电影的信息，使用data.frame()函数创建，其中每个元素为一个向量：

```
> movies <- data.frame(name = c("Toy Story","Akira","Fight Club","City of God"),year = c(1995,1998,1999,2002))
> movies
        name year
1   Toy Story 1995
2       Akira 1998
3  Fight Club 1999
4 City of God 2002
```

访问数据框中的列元素有两种方法，通过元素名字索引或通过元素位置索引进行访问，如下图所示：

```
> movies$name
[1] Toy Story   Akira       Fight Club  City of God
Levels: Akira City of God Fight Club Toy Story
> movies[1]
         name
1   Toy Story
2       Akira
3  Fight Club
4 City of God
```

查看数据框的头部和尾部，可用 head()和 tail()函数实现，分别显示六个元素。此外，向数据框中插入新的行和列元素，可通过一下代码实现，其中，插入列元素的方法与列表类似，用 rbind()函数插入行元素，如下图所示：

1.5 R 的基本编程原理

每种编程语言都会有条件、循环等基本操作，R 作为一种擅长统计分析的语言，通常会应用条件判断与循环来实现对数据的处理，本节中将会介绍 R 中的条件语句和循环语句，R 中的函数及对象和类的相关概念。

1.5.1 条件语句

R 中的条件语句与其他编程语言类似，通过 if（条件）进行判断，如条件为真，则执行 if 后的语句，否则执行 else 后的语句，其基本操作如下：

```
> a <- 10
> if(a>5){
+   print('greater than 5')
+ }else{
+   print('not greater than 5')
+ }
[1] "greater than 5"
```

在条件语句中，总会接触到逻辑算符，下面为 R 中常见的逻辑算符：

算符	意义
==	等于
!=	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于
&	并且
|	或
!	非
%in%	前项是否在后项中

1.5.2 循环语句

在 R 中，常用的循环有 for 循环和 while 循环。for 循环可通过向量中所有的值，如下图示例，将会按照顺序输出向量中的每个值。

```
> years <- c(1909,1953,1997,2009,2017)
> for(yr in years){
+   print(yr)
+ }
[1] 1909
[1] 1953
[1] 1997
[1] 2009
[1] 2017
```

每次循环，都用定义的 yr 变量访问向量 year，直到向量中的元素全部被输出，循环结束。

while 循环与 for 不同，只要 while 循环中的条件正确，循环就会一直被执行，如下面示例中的 yr 变量，它控制着循环是否进行，只要 yr 符合条件，循环就会一直进行，直到 yr=2016 时跳出循环。

```
> yr <- 2010
> while(yr<2016){
+   print(yr)
+   yr <- yr+1
+ }
[1] 2010
[1] 2011
[1] 2012
[1] 2013
[1] 2014
[1] 2015
```

1.5.3R 中的函数

R 中的函数有两种：预定义函数和自定义函数，我们在前面的实例中已经对预定义函数有所涉及，如 mean()函数用于求均值，sum()函数用于求和等，由于在现实操作中，预定义函数会伴随数据分析的全过程，在此不做赘述。

在 R 中通过 function 实现对自定义函数的定义，如下示例定义了一个在屏幕上输出“Hello World”语句的函数：

```
> printHelloWorld<-function(){
+   print("Hello World")
+ }
> printHelloWorld()
[1] "Hello World"
```

也可以使用显示的返回值，用 return()实现：

```
> sum<-function(x,y){
+   return(x+y)
+ }
> sum(1,2)
[1] 3
```

下面的示例中定义了 x 和 s 两个变量，其中 x 的值为 3.14，s 的值为“Hello World”字符串，在函数外，输入 x 返回 3.14，输入 s 时，则会报错，这就是我们常说的全局变量和局部变量，观察变量定义语句我们可以发现，用“<<-”来定义全局变量，而赋值号则用来定义局部变量。

```
> Function<-function(){
+   x <<- 3.14
+   s <- 'Hello World'
+   return(s)
+ }
> Function()
[1] "Hello World"
> x
[1] 3.14
> s
Error: object 's' not found
```