3. Time Series Analysis

在R中，日期数据别单独归为一个Date类。一般用整数保存，数值为从1970-1-1 经过的天数。R中用一种叫做POSIXct和POSIXlt的特殊数据类型保存日期和时间，可以仅包含日期部分，也可以同时有日期和时间。技术上， POSIXct把日期时间保存为从1970年1月1日零时到该日期时间的时间间隔秒数。日期时间会涉及到所在时区、夏时制等问题，比较复杂。

```
> Sys.Date()#查看当前系统的时间
> class(Sys.Date())#查看数据类型
```

- as.Date()函数：将数据转换为日期数据，使用format参数决定外观。 %Y表示四位数年份（y表示两位数年份），%m表示月份，%d表示日期

```
> a <- c("240228", "2024/3/1", "2024:2:30")
> b <- as.Date(a[1], format = "%y%m%d") #将数据转换为日期数据
> class(a)
[1] "character"
> class(b)
[1] "Date"
> b
[1] "2024-02-28"
> as.Date(a[2], format = "%Y/%m/%d")
[1] "2024-03-01"
> as.Date(a[3], format = "%Y:%m:%d")
[1] NA
```

- lubridate是一个提供了许多日期和日期时间功能的扩展包。

```
> library(lubridate)
> class("2024-03-18")
[1] "character"
> lubridate::ymd("2024-03-18")
[1] "2024-03-18"
> class(lubridate::ymd("2024-03-18"))
[1] "Date"
> b <- lubridate::make_date(2024,5,20)
> b
[1] "2024-05-20"
> class(b)
[1] "Date"
> year(b)
[1] 2024
> month(b)
[1] 5
> day(b)
[1] 20
```

3.1 生成时间序列数据

R 软件中有些时间序列分析函数需要特定的时间序列类型数据作为输入。常用的有 ts 类型、xts类型等。ts类型用于保存一元或者多元的等间隔时间序列，如月度、季度、年度数据。

- seq()函数：创建连续的时间点,要使用as.Date()系统才会当做时间数据进行处理。

```
> seq(1,10,2) #创建自然数序列
> seq(as.Date("2022-01-10"),as.Date("2022-03-08"),by=10)
[1] 1 3 5 7 9
[1] "2022-01-10" "2022-01-20" "2022-01-30" "2022-02-09" "2022-02-19"
[6] "2022-03-01"
```

- ts()函数：把向量转化为时间序列数据。start()和end()返回起始点的时间，frequency()返回频率：1为年度数据，4为季度数据，12为月度数据

```
> sales <- seq(100,200,5)
> ts(sales,start=c(2000,2),frequency = 1) #年度数据
Time Series:
Start = 2001
End = 2021
Frequency = 1
 [1] 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190
[20] 195 200
> ts(sales,start=c(2000,2),frequency = 12) #月度数据
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2000     100 105 110 115 120 125 130 135 140 145 150
2001 155 160 165 170 175 180 185 190 195 200
> ts(sales,start=c(2000,2),frequency = 4)  #季度数据
     Qtr1 Qtr2 Qtr3 Qtr4
2000       100  105  110
2001  115  120  125  130
2002  135  140  145  150
2003  155  160  165  170
2004  175  180  185  190
2005  195  200
```

3.2 读入时间序列数据

常见的数据形式是用文本格式保存，数据之间用空行、空格或者逗号分隔。从文件读入时间序列数据：

- 单个向量读入：数据仅包含序列的数值而不包含时间标签，这时数据可以保存为用空行和空格分隔的文本格式数据，例如文件“sp500-2691.txt”是标准普尔500指数月超额收益率从1926年1月到1991年12月的数据。

```
> x <- scan("myData/sp500-2691.txt",skip=1)
> head(x)
[1]  0.0225 -0.0440 -0.0591  0.0227  0.0077  0.0432
> sp500 <- ts(x,start=c(1926,1),frequency=12)
> start(sp500)
[1] 1926    1
> end(sp500)
[1] 1991   12
> frequency(sp500)
[1] 12
> length(sp500)
[1] 792
> head(as.Date(time(sp500)),20)
 [1] "1926-01-01" "1926-02-01" "1926-03-01" "1926-04-01" "1926-05-01"
 [6] "1926-06-01" "1926-07-01" "1926-08-01" "1926-09-01" "1926-10-01"
[11] "1926-11-01" "1926-12-01" "1927-01-01" "1927-02-01" "1927-03-01"
[16] "1927-04-01" "1927-05-01" "1927-06-01" "1927-07-01" "1927-08-01"
> head(season(sp500),20)
 [1] January   February  March     April     May       June      July
 [8] August    September October   November  December  January   February
[15] March     April     May       June      July      August
12 Levels: January February March April May June July August ... December
```

- 带有时间标签的序列：同一行的数据之间用空格或者制表符分隔。文件“d-ibm-0110.txt”为IBM股票从2001-1-2到2010-12-31的日简单收益率。

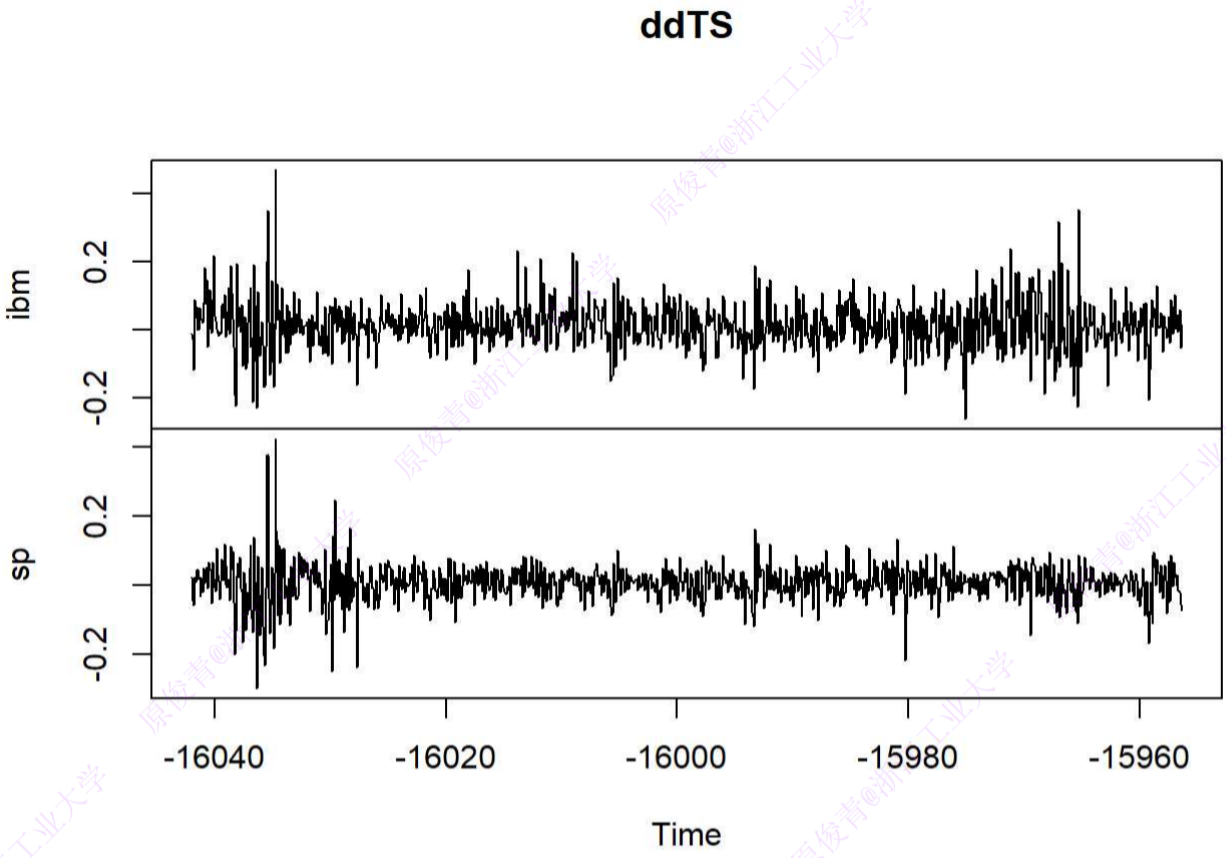
```
> ##读入R数据框
> d <-read.table(
+ "myData/d-ibm-0110.txt", header=TRUE, colClasses=c ("character","numeric"))
> head(d,10)
       date    return
1  20010102 -0.002206
2  20010103  0.115696
3  20010104 -0.015192
4  20010105  0.008719
5  20010108 -0.004654
6  20010109 -0.010688
7  20010110  0.009453
8  20010111  0.002676
9  20010112  0.001334
10 20010116 -0.011326
> #其中的日期可以用如下程序转换成 R 的 Date 类型：
> d[["date"]]<-lubridate::ymd(d[[ "date"]])
> head(d,10)
         date    return
1  2001-01-02 -0.002206
2  2001-01-03  0.115696
3  2001-01-04 -0.015192
4  2001-01-05  0.008719
5  2001-01-08 -0.004654
6  2001-01-09 -0.010688
7  2001-01-10  0.009453
8  2001-01-11  0.002676
9  2001-01-12  0.001334
10 2001-01-16 -0.011326
```

- 有些序列的年月日是分开输入的，比如，文件“m-unrate.txt”为美国从1948年1月到2010年9月的经季节调整的月失业率百分数。

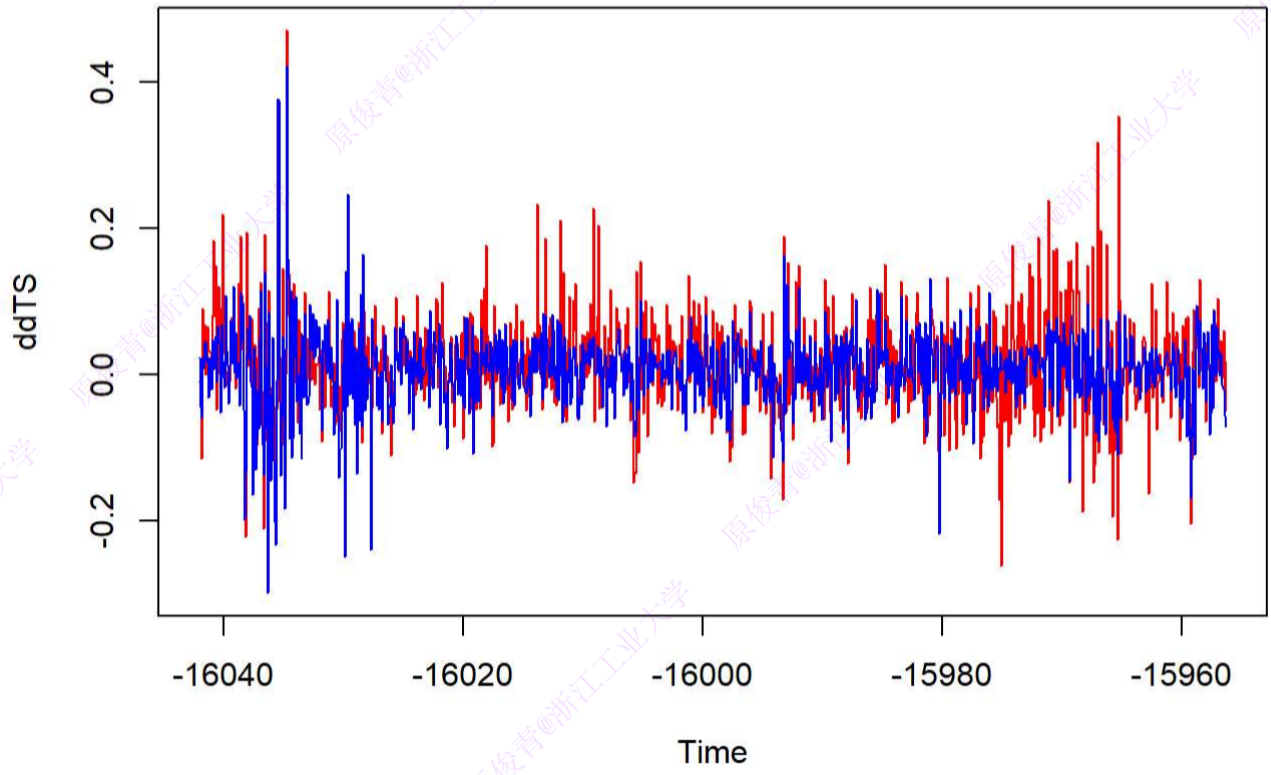
```
> d <- read.table("myData/m-unrate.txt",  header=TRUE, colClasses=rep("numeric" , 4))
> head(d)
  Year mon dd rate
1 1948   1  1  3.4
2 1948   2  1  3.8
3 1948   3  1  4.0
4 1948   4  1  3.9
5 1948   5  1  3.5
6 1948   6  1  3.6
> ddts <- ts(d[["rate"]], start=c(1948,1), frequency=12)
> end(ddts)
[1] 2010    9
> length(ddts)
[1] 753
>
> d[["date"]] <- make_date(d[["Year"]],d[["mon"]],d[["dd"]])
> head(d)
  Year mon dd rate       date
1 1948   1  1  3.4 1948-01-01
2 1948   2  1  3.8 1948-02-01
3 1948   3  1  4.0 1948-03-01
4 1948   4  1  3.9 1948-04-01
5 1948   5  1  3.5 1948-05-01
6 1948   6  1  3.6 1948-06-01
```

- CSV 文件也是一种文本格式的数据文件，相对而言更规范一些，所以经常用于在不同的数据管理和数据分析软件之间传递数据。例如："m-ibmsp-2611.csv"文件为IBM股票和标准普尔500指数的月超额收益率从1926年1月到2011年9月的数据。window()可以取出时间序列的一段，ts.intersect()和ts.union()可以形成多元时间序列。

```
> library(readr)
> d <- read_csv("myData/m-ibmsp-2611.csv",col_types=cols(date=col_date(format="%Y%m%d"), .defau
lt=col_double()))
> dim(d)
[1] 1029    3
> head(d)
# A tibble: 6 × 3
  date            ibm       sp
  <date>        <dbl>    <dbl>
1 1926-01-30 -0.0104  0.0225
2 1926-02-27 -0.0245 -0.0440
3 1926-03-31 -0.116  -0.0591
4 1926-04-30  0.0898  0.0227
5 1926-05-28  0.0369  0.00768
6 1926-06-30  0.0685  0.0432
> tail(d)
# A tibble: 6 × 3
  date            ibm      sp
  <date>        <dbl>   <dbl>
1 2011-04-29  0.0461   0.0285
2 2011-05-31 -0.00528 -0.0135
3 2011-06-30  0.0155  -0.0183
4 2011-07-29  0.0600  -0.0215
5 2011-08-31 -0.0505  -0.0568
6 2011-09-30  0.0172  -0.0718
> ddTS <- ts(d[2:3], start=d[1,1], frequency=12) #第2列第3列
> head(ddTS,10)
             ibm         sp
 [1,] -0.010381  0.022472
 [2,] -0.024476 -0.043956
 [3,] -0.115591 -0.059113
 [4,]  0.089783  0.022688
 [5,]  0.036932  0.007679
 [6,]  0.068493  0.043184
 [7,]  0.000000  0.045455
 [8,]  0.000000  0.017081
 [9,]  0.065104  0.022901
[10,]  0.032258 -0.031343
> plot(ddTS) ##每个序列单独画一个窗格
```




```
> ##col指定颜色，lty指定线型，lwd指定线粗细，pch指定散点类型，cex指定散点大小
> plot(ddTS, plot.type="single", col=c("red", "blue"))  ##画在同一坐标系中
```



```
>
>
>
> ibmTS <- ts(d[["ibm"]], start=c(1926,1), frequency=12)
> spTS <- ts(d[["sp"]], start=c(1926,1), frequency=12)
> start(ibmTS);end(ibmTS)
[1] 1926    1
[1] 2011    9
> ibm <- window(ibmTS,start=c(2000,1),end=c(2010,12),frequency=4)
> sp <- window(spTS,start=c(1998,1),end=c(2005,12),frequency=4)
> ts.union(ibm,sp)
               ibm         sp
1998 Q1         NA  0.010150
1998 Q2         NA  0.009076
1998 Q3         NA -0.011615
1998 Q4         NA  0.080294
1999 Q1         NA  0.041009
1999 Q2         NA  0.037944
1999 Q3         NA -0.032046
1999 Q4         NA  0.062539
2000 Q1  0.040556 -0.050904
2000 Q2 -0.055085 -0.030796
2000 Q3  0.026241 -0.016341
2000 Q4 -0.124444 -0.004949
2001 Q1  0.317647  0.034637
2001 Q2  0.197130  0.076814
2001 Q3 -0.068938 -0.010772
2001 Q4  0.178260  0.018099
2002 Q1 -0.108052 -0.015574
2002 Q2 -0.194615 -0.061418
2002 Q3 -0.022222 -0.078995
2002 Q4  0.353799  0.086436
2003 Q1  0.009032 -0.027415
2003 Q2  0.082494  0.081044
2003 Q3 -0.015152  0.016224
2003 Q4  0.013019  0.054962
2004 Q1  0.070673  0.017276
2004 Q2 -0.039961 -0.016791
2004 Q3 -0.012252 -0.034291
2004 Q4  0.046769  0.014014
2005 Q1 -0.052343 -0.025290
2005 Q2 -0.164150 -0.020109
2005 Q3  0.124798  0.035968
2005 Q4  0.020693 -0.017741
2006 Q1 -0.010949        NA
2006 Q2 -0.001576        NA
2006 Q3  0.007680        NA
2006 Q4  0.126800        NA
2007 Q1  0.020587        NA
2007 Q2  0.084341        NA
2007 Q3  0.051306        NA
2007 Q4 -0.014261        NA
2008 Q1 -0.009158        NA
2008 Q2  0.048289        NA
2008 Q3  0.079727        NA
```

```
2008 Q4 -0.205113         NA
2009 Q1  0.088997         NA
2009 Q2  0.065229         NA
2009 Q3  0.129381         NA
2009 Q4  0.008361         NA
2010 Q1 -0.065011         NA
2010 Q2  0.005848         NA
2010 Q3  0.039844         NA
2010 Q4  0.070523         NA
> ts.intersect(ibm,sp)
               ibm         sp
2000 Q1  0.040556 -0.050904
2000 Q2 -0.055085 -0.030796
2000 Q3  0.026241 -0.016341
2000 Q4 -0.124444 -0.004949
2001 Q1  0.317647  0.034637
2001 Q2  0.197130  0.076814
2001 Q3 -0.068938 -0.010772
2001 Q4  0.178260  0.018099
2002 Q1 -0.108052 -0.015574
2002 Q2 -0.194615 -0.061418
2002 Q3 -0.022222 -0.078995
2002 Q4  0.353799  0.086436
2003 Q1  0.009032 -0.027415
2003 Q2  0.082494  0.081044
2003 Q3 -0.015152  0.016224
2003 Q4  0.013019  0.054962
2004 Q1  0.070673  0.017276
2004 Q2 -0.039961 -0.016791
2004 Q3 -0.012252 -0.034291
2004 Q4  0.046769  0.014014
2005 Q1 -0.052343 -0.025290
2005 Q2 -0.164150 -0.020109
2005 Q3  0.124798  0.035968
2005 Q4  0.020693 -0.017741
```

- xts也是一种时间序列数据类型，既可以保存等间隔时间序列数据，也可以保存不等间隔的时间序列数据，并且xts类型的数据访问功能更为方便。读入方法例如xts(x,date),其中x是向量、矩阵或数据框，date是日期或者日期时间。x取矩阵或者数据框时每列是一个时间序列。有了 xts 类型的变量 x 后，可以用coredata(x)返回x的不包含时间的纯数据；用index(x)返回 x的时间标签。例如，文件“d-ibm-0110.txt”为IBM股票从2001-1-2到2010-12-31的日简单收益率，读入数据并转换为xts类型的程序如下：

```
> library(xts)
> d <-read.table(
+ "myData/d-ibm-0110.txt", header=TRUE, colClasses=c ("character","numeric"))
> head(d,10)
       date    return
1  20010102 -0.002206
2  20010103  0.115696
3  20010104 -0.015192
4  20010105  0.008719
5  20010108 -0.004654
6  20010109 -0.010688
7  20010110  0.009453
8  20010111  0.002676
9  20010112  0.001334
10 20010116 -0.011326
> ibmrtn <- xts(d[["return"]],lubridate::ymd(d[[ "date"]]))
> class(ibmrtn)
[1] "xts" "zoo"
> head(ibmrtn,10)
             [,1]
2001-01-02 -0.002206
2001-01-03  0.115696
2001-01-04 -0.015192
2001-01-05  0.008719
2001-01-08 -0.004654
2001-01-09 -0.010688
2001-01-10  0.009453
2001-01-11  0.002676
2001-01-12  0.001334
2001-01-16 -0.011326
> head(coredata(ibmrtn),10) #不包含时间的纯数据
           [,1]
 [1,] -0.002206
 [2,]  0.115696
 [3,] -0.015192
 [4,]  0.008719
 [5,] -0.004654
 [6,] -0.010688
 [7,]  0.009453
 [8,]  0.002676
 [9,]  0.001334
[10,] -0.011326
> head(index(ibmrtn),10) #时间标签
 [1] "2001-01-02" "2001-01-03" "2001-01-04" "2001-01-05" "2001-01-08"
 [6] "2001-01-09" "2001-01-10" "2001-01-11" "2001-01-12" "2001-01-16"
```

- 对于ts类型的时间序列，time()返回每个时间点的时间，cycle()返回每个数据点的月份（数值型），season()返回每个数据点的月份（字符型）。

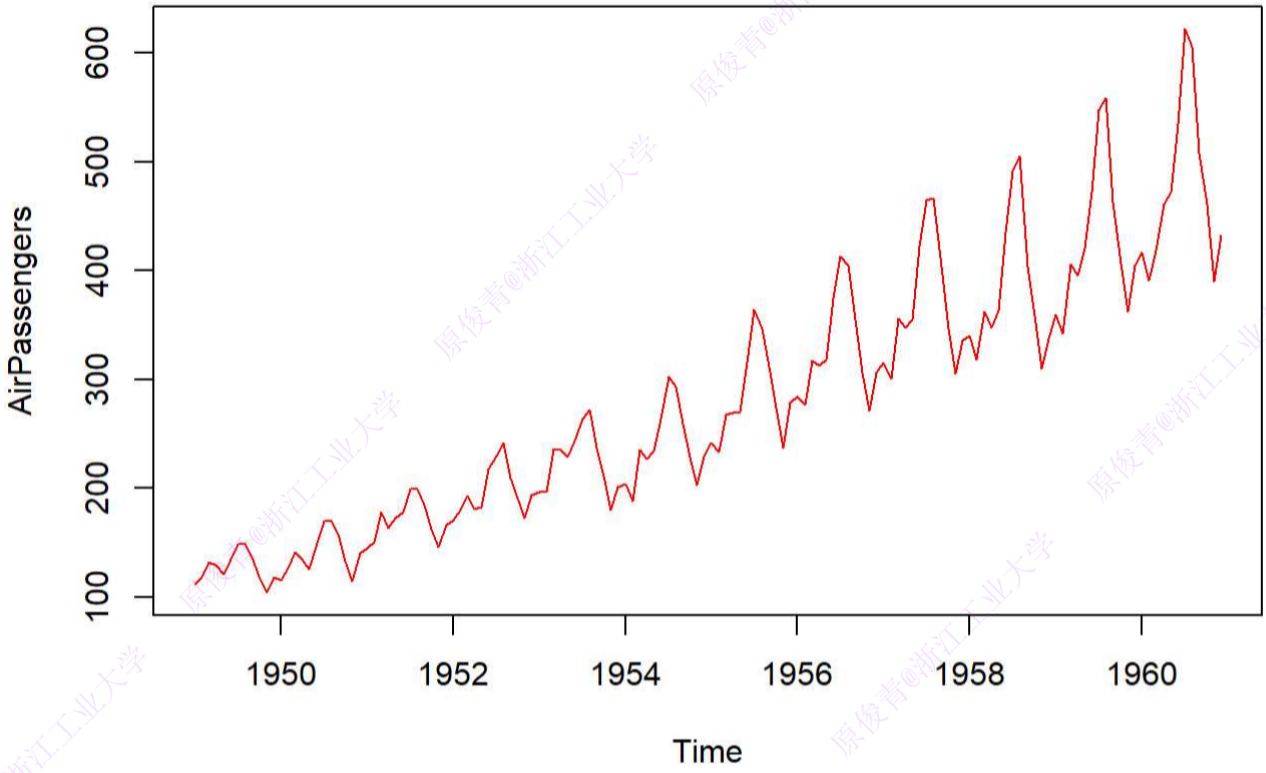
```
> head(AirPassengers)
[1] 112 118 132 129 121 135
> class(AirPassengers)
[1] "ts"
> head(time(AirPassengers),10)
 [1] 1949.000 1949.083 1949.167 1949.250 1949.333 1949.417 1949.500 1949.583
 [9] 1949.667 1949.750
> head(cycle(AirPassengers),10)
 [1]  1  2  3  4  5  6  7  8  9 10
> head(season(AirPassengers),10)
 [1] January   February  March     April     May       June      July
 [8] August    September October
12 Levels: January February March April May June July August ... December
> class(season(AirPassengers))  ##factor
[1] "factor"
>
> start(AirPassengers)
[1] 1949    1
> end(AirPassengers)
[1] 1960   12
> frequency(AirPassengers)
[1] 12
> length(AirPassengers)
[1] 144
>
> window(AirPassengers,start=c(1950,1),end=c(1952,12))
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
> window(AirPassengers,start=c(1950,1),end=c(1958,12),frequency=4)
     Qtr1 Qtr2 Qtr3 Qtr4
1950  115  135  170  133
1951  145  163  199  162
1952  171  181  230  191
1953  196  235  264  211
1954  204  227  302  229
1955  242  269  364  274
1956  284  313  413  306
1957  315  348  465  347
1958  340  348  491  359
```

- as.xts()转化为xts类型的时间序列

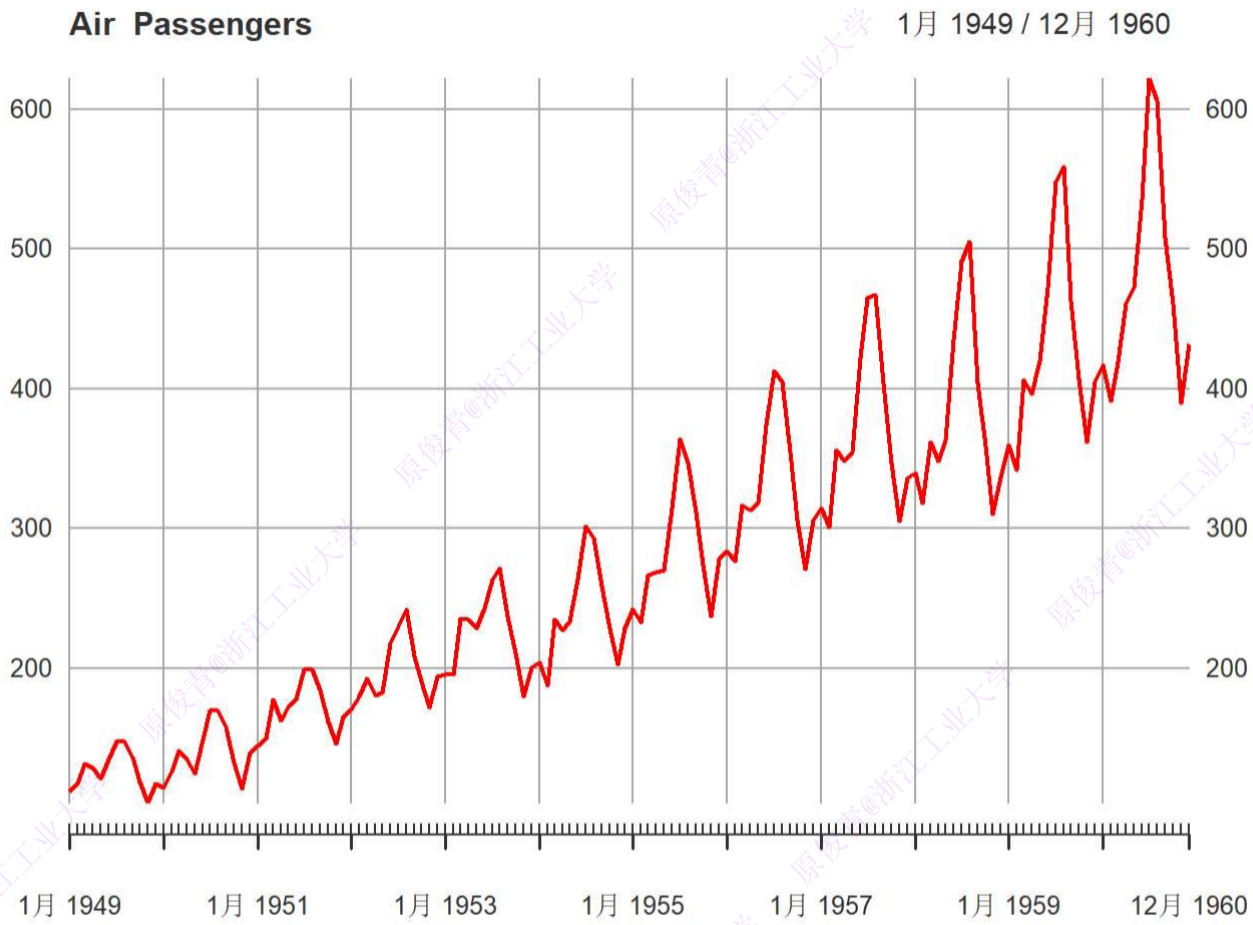
```
> library(xts)
> xts.ap <-as.xts(AirPassengers)
> head(xts.ap,10)
          [,1]
1月 1949   112
2月 1949   118
3月 1949   132
4月 1949   129
5月 1949   121
6月 1949   135
7月 1949   148
8月 1949   148
9月 1949   136
10月 1949  119
> head(coredata(xts.ap),10)
      [,1]
 [1,]  112
 [2,]  118
 [3,]  132
 [4,]  129
 [5,]  121
 [6,]  135
 [7,]  148
 [8,]  148
 [9,]  136
[10,]  119
> head(index(xts.ap),10)
 [1] "1月 1949"  "2月 1949"  "3月 1949"  "4月 1949"  "5月 1949"  "6月 1949"
 [7] "7月 1949"  "8月 1949"  "9月 1949"  "10月 1949"
> class(index(xts.ap))  ##yearmon数据类型
[1] "yearmon"
> head(date(xts.ap),10)
 [1] "1949-01-01" "1949-02-01" "1949-03-01" "1949-04-01" "1949-05-01"
 [6] "1949-06-01" "1949-07-01" "1949-08-01" "1949-09-01" "1949-10-01"
> head(year(xts.ap),10)
 [1] 1949 1949 1949 1949 1949 1949 1949 1949 1949 1949
> head(month(xts.ap),10)
 [1]  1  2  3  4  5  6  7  8  9 10
> head(day(xts.ap),10)
 [1] 1 1 1 1 1 1 1 1 1 1
```

- 对ts类型和xts类型数据作图。

```
> plot(AirPassengers, col="red")
```



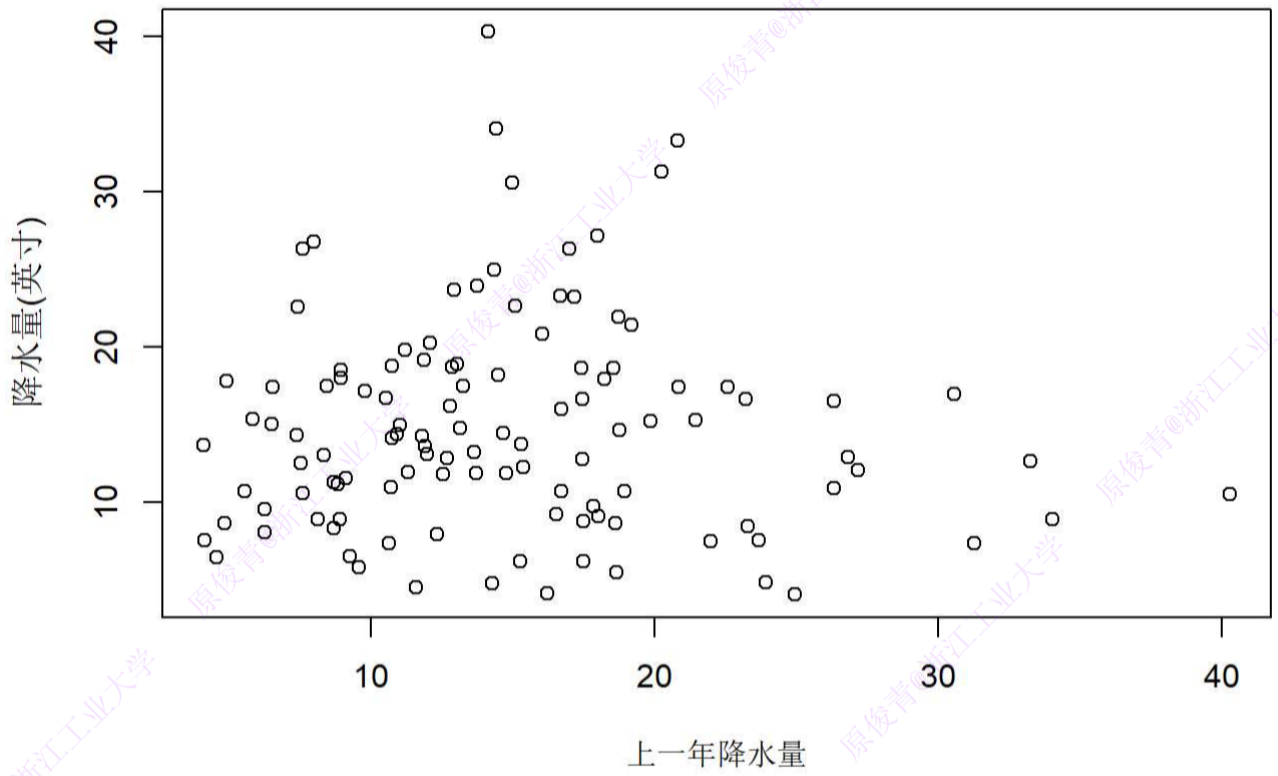

```
> plot (xts.ap, main="Air  Passengers", major.ticks="year", minor.ticks=NULL, grid.ticks.on="year", col="red")
```



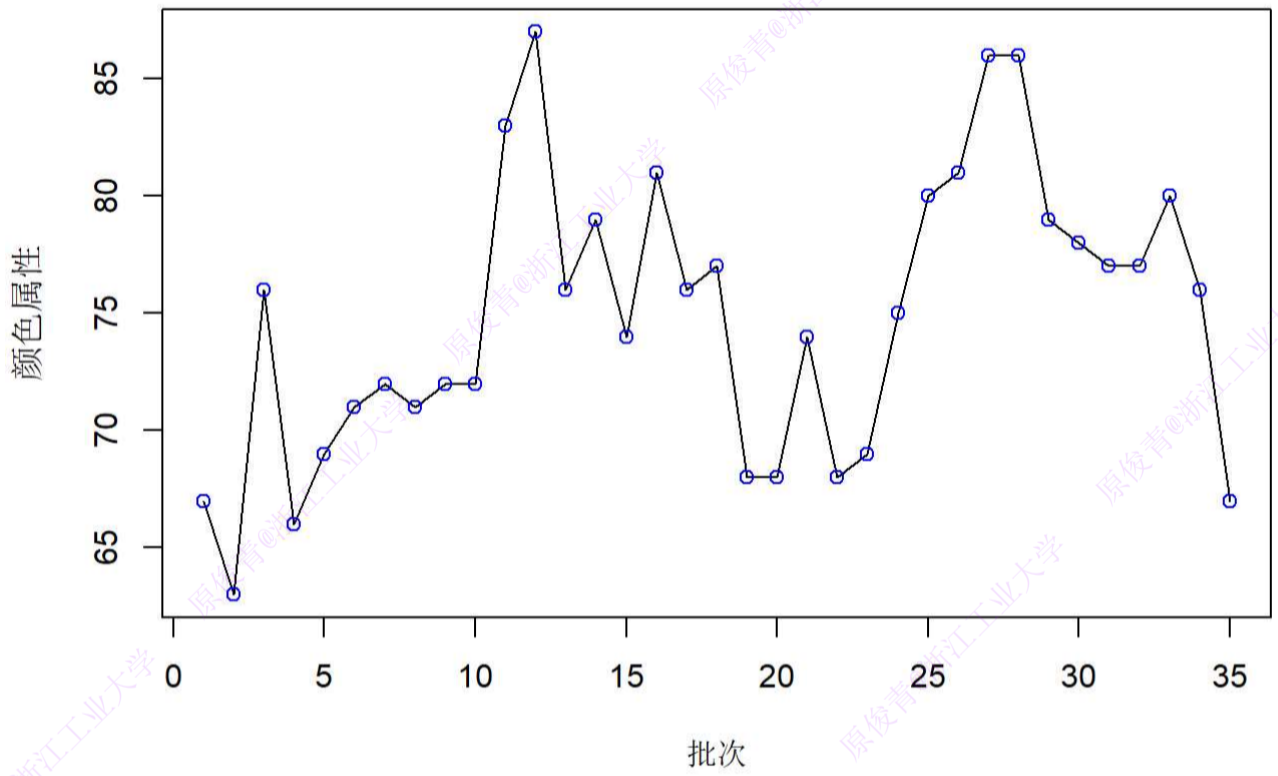
习题1. (on P.7)

Question 1~2.

```
> # Question. 1.1
> library(TSA); data(larain)
> #dev.new()
> #win.graph(width=3,height=3,pointsize=8)
> plot(y=larain,x=zlag(larain),ylab='降水量(英寸)',xlab='上一年降水量')
```

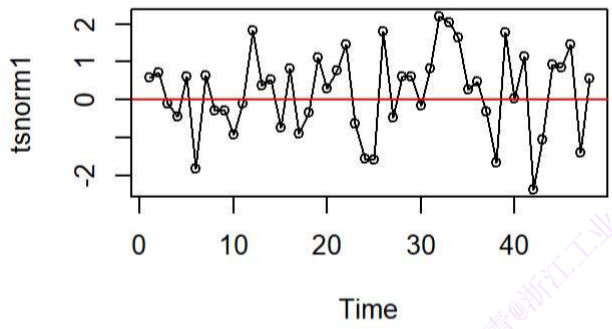


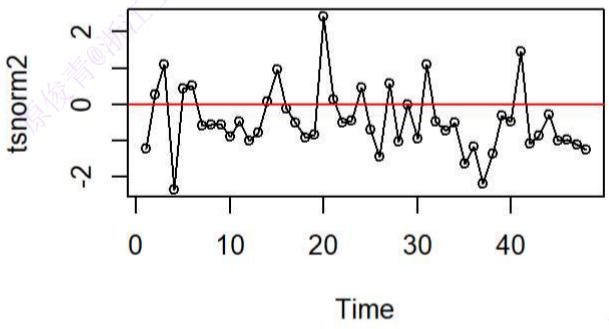

```
> # Question. 1.2
> library(TSA)
> data(color)
> #dev.new()
> #win.graph(width=3,height=3,pointsize=8)
> plot(color,ylab='颜色属性',xlab='批次',type='l')
> points(color,col='blue')
```

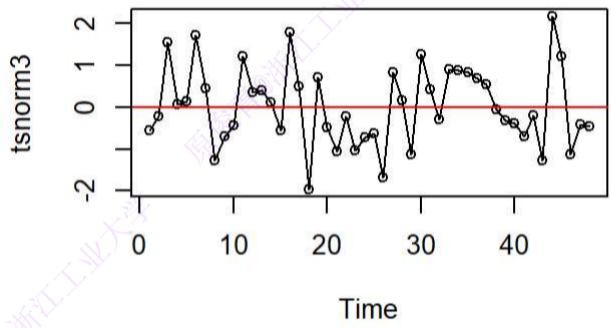


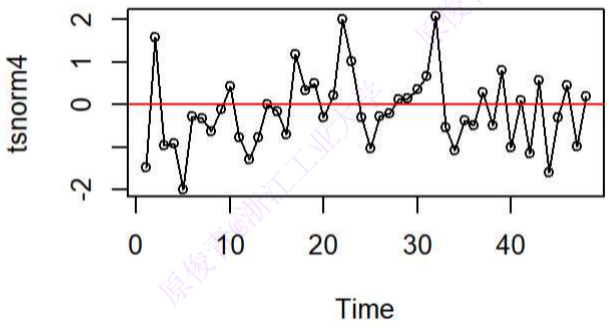
Question 3~5.

```
> ###  rnorm(n), rchisq(n,df=2), rt(n,df=5)
> n=48;
> set.seed(12345); tsnorm1 = ts(rnorm(n),freq=1,start=1)
> set.seed(1234); tsnorm2 = ts(rnorm(n),freq=1,start=1)
> set.seed(123); tsnorm3 = ts(rnorm(n),freq=1,start=1)
> set.seed(12); tsnorm4 = ts(rnorm(n),freq=1,start=1)
> opar=par(mfrow=c(2,2))
> plot(tsnorm1,type='o'); abline(h=0,col='red')
> plot(tsnorm2,type='o'); abline(h=0,col='red')
> plot(tsnorm3,type='o'); abline(h=0,col='red')
> plot(tsnorm4,type='o'); abline(h=0,col='red')
```





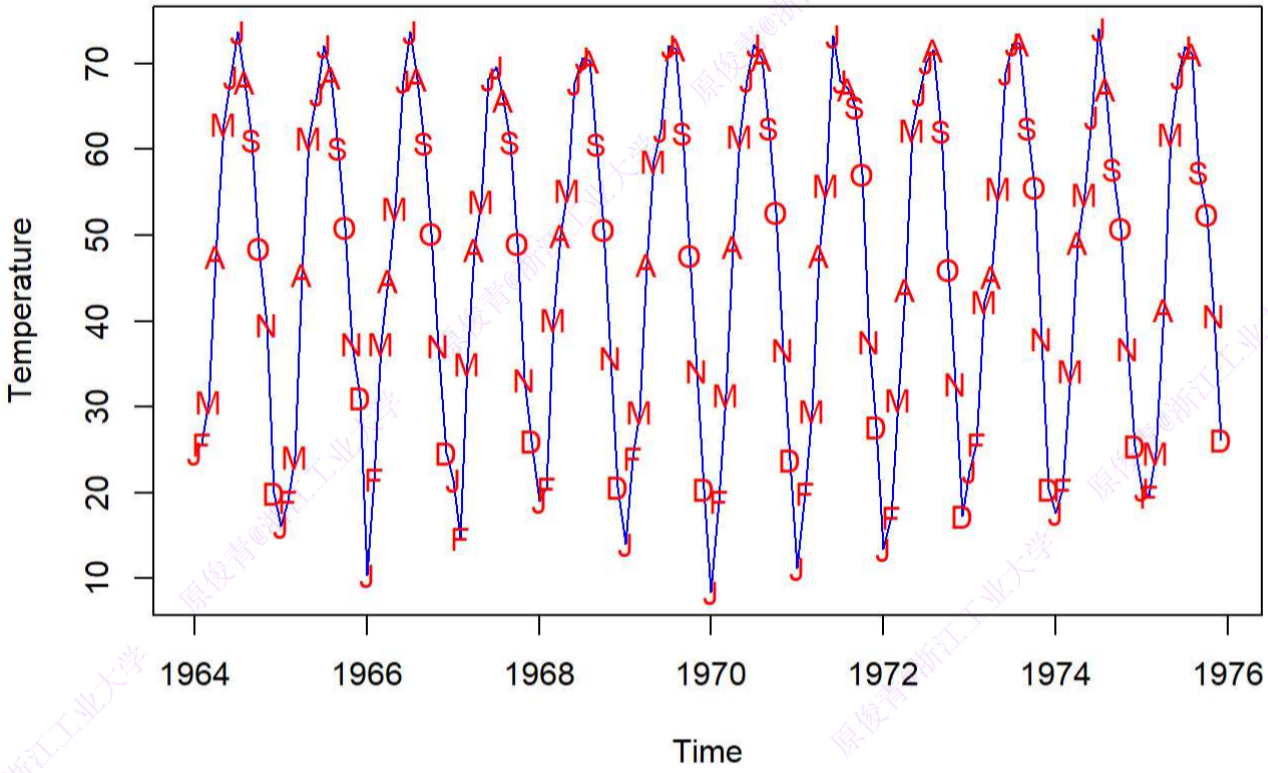





```
> par(opar)
```

Question 6.

```
> library(TSA)
> data(tempdub)
> #op <- par(bg = "light blue")
> plot(tempdub,ylab='Temperature',type='l',col = "blue")
> Month=c("J","F","M","A","M","J","J","A","S","O","N","D")
> points(tempdub,pch=Month,col='red')
```




```
> #points(tempdub,pch=as.vector(season(tempdub)),col='red')
> #par(op)
```