金融时间序列分析常见问题集

邓一硕

目录

1 前言

本书分为上下两篇，上篇为基础篇，下面为应用篇。

2 基础篇

2.1 时间序列基础

2.1.1 创建时间序列对象

需要加载的 R 包

2.1.1.1 R 进行金融分析时最常用的时间序列类型是什么? zoo 包中的 zoo 格式，xts 包中的 xts 格式和 timeSeries 包中的 timeSeries 格式，使用 timeSeries 包时需要 timeDate 提供支撑。

2.1.1.2 这三种时间序列对象的时间戳有哪些不同? zoo 类型时间序列对象和 xts 类型的时间序列对象的时间戳标记取决于生成时间序列对象是所使用的时间戳标记的类型。而 timeSeries 对象的时间戳标记与生成时间序列对象时所使用的时间戳标记是独立的，timeSeries 对象的时间戳通常是数值型。

2.1.1.3 创建时间序列对象时最常用的时间戳标记是什么类型的? 对于 zoo 对象和 xts 对象而言，当其处理的是日记录数据时，由于不用关心时区信息，因此通常用 Date 类型的时间戳标记；当期处理的是盘中数据时、或者涉及时区信息和夏时令时，通常用 POSIXct 类型的时间戳标记。timeSeries 对象的时间戳标记通常是 timeDate 对象，timeDate 本身携带了 Olsen 时区数据的基准信息。

2.1.1.4 时间序列对象依赖于其被创建的方式吗? 对于依赖于操作系统时区信息的 zoo 对象和 xts 对象而言是这样的，zoo 对象和 xts 对象的显示结果跟操作系统的内部时区设置以及夏令时规则有关，因此，同样的 zoo 对

象和 xts 对象在不同的操作系统上可能结果不一致。而 timeSeries 对象不受此影响，因为 timeSeries 对象的时间戳标识为 timeDate 对象。timeDate 对象一般是以 POSIXct 格式存储的 GMT 时间，其将时区和 DST 信息与 Rmetrics 中的 Olsens 时区数据库单独存储。

2.1.1.5 在不需要关心时区信息时，当前的时区环境对时间序列对象的创建有什么影响? 对于 zoo 对象和 xts 对象，作为默认时间戳标识类型的 as.POSIXct 函数会基于本地系统环境中的时区设置来创建时间戳。

```
options(width = 50)
args(zoo)
```

```
## function (x = NULL, order.by = index(x), frequency = NULL, calendar = getOption("zoo
##     TRUE))
## NULL
```

```
args(xts)
```

```
## function (x = NULL, order.by = index(x), frequency = NULL, unique = TRUE,
##     tzone = Sys.getenv("TZ"), ...)
## NULL
```

```
args(as.POSIXct)
```

```
## function (x, tz = "", ...)
## NULL
```

因此，虽然操作相同，伦敦、纽约和东京的用户却将得到不同的操作结果。采用 ISOdatatime 函数创建时间戳时也会遇到同样的情况。

```
options(width = 50)
args(ISOdatetime)
```

```
## function (year, month, day, hour, min, sec, tz = "")
## NULL
```

慎用 ISOdate 函数!

```
options(width = 50)
args(ISOdate)
```

```
## function (year, month, day, hour = 12, min = 0, sec = 0, tz = "GMT")
## NULL
```

默认设置下，ISOdate 函数会创建一个基于 GMT 时间的时间戳标识，只是时间提前了 12 小时。

注意: 三个函数都将返回一个 POSIXct 类型的时间戳标识。对于 timeSeries 对象而言，当基于字符串、合适的时区信息或者标识金融中心信息时都无需考虑系统环境问题。

```
options(width = 50)
args(timeSeries)
```

```
## function (data, charvec, units = NULL, format = NULL, zone = "",
##     FinCenter = "", recordIDs = data.frame(), title = NULL, documentation = NULL,
##     ...)
## NULL
```

```
args(timeDate)
```

```
## function (charvec, format = NULL, zone = "", FinCenter = "",
##     ...)
## NULL
```

2.1.1.6 如何查看当前系统的时区环境? Sys.timezone() 函数可以返回系统中时区环境的当前设置。

```
Sys.timezone()
```

```
## [1] "Asia/Shanghai"
```

2.1.1.7 如何基于现有的字符格式的时间戳创建日时间序列对象的时间戳标识? 对于 zoo 对象和 xts 对象，创建日时间序列数据的时间戳标识可以用 Date 函数。

```
args(as.Date)
```

```
## function (x, ...)
## NULL
```

Date 对象不涉及时区信息，系统一般讲 Date 对象默认为 GMT 时间格式。对于 timeSeries 对象，最好把现有的文本型时间戳调整为 ans 格式。

常规数据

```
set.seed(1953)
data <- rnorm(6)
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
zoo(data, as.Date(charvec))
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.0219246 -0.9043255  0.4132378  0.1866212
## 2009-05-01 2009-06-01
##  0.2308177  0.2356802
```

xts:

```
xts(data, as.Date(charvec))
```

```
##                  [,1]
## 2009-01-01  0.0219246
## 2009-02-01 -0.9043255
## 2009-03-01  0.4132378
## 2009-04-01  0.1866212
## 2009-05-01  0.2308177
## 2009-06-01  0.2356802
```

timeSeries:

```
timeSeries(data, charvec)
```

```
## GMT
##                  TS.1
## 2009-01-01  0.0219246
## 2009-02-01 -0.9043255
## 2009-03-01  0.4132378
## 2009-04-01  0.1866212
## 2009-05-01  0.2308177
## 2009-06-01  0.2356802
```

2.1.1.8 如何创建过去 50 天的日时间序列数据?

常规数据:

```
set.seed(1953)
data <- matrix(rnorm(22), ncol = 2)
now <- "2009-01-05"
```

zoo:

```
zoo(data, as.Date(now) - 0:10)
```

```
##
## 2008-12-26 -0.132607774  0.47362237
## 2008-12-27 -1.421197846  0.20158998
## 2008-12-28 -0.004685482 -0.25997610
## 2008-12-29  1.099462672  2.50495921
## 2008-12-30  1.483738895  0.57023860
## 2008-12-31  0.235680176  0.79178150
## 2009-01-01  0.230817687 -1.51775046
## 2009-01-02  0.186621223 -0.52321193
## 2009-01-03  0.413237769 -2.33069954
## 2009-01-04 -0.904325473 -0.07551388
## 2009-01-05  0.021924601  0.16479626
```

xts:

```
xts(data, as.Date(now) - 0:10)
```

```
##                    [,1]        [,2]
## 2008-12-26 -0.132607774  0.47362237
## 2008-12-27 -1.421197846  0.20158998
## 2008-12-28 -0.004685482 -0.25997610
## 2008-12-29  1.099462672  2.50495921
## 2008-12-30  1.483738895  0.57023860
## 2008-12-31  0.235680176  0.79178150
## 2009-01-01  0.230817687 -1.51775046
## 2009-01-02  0.186621223 -0.52321193
## 2009-01-03  0.413237769 -2.33069954
## 2009-01-04 -0.904325473 -0.07551388
## 2009-01-05  0.021924601  0.16479626
```

timeSeries:

```
timeSeries(data, as.Date(now) - 0:10)
```

```
## GMT
##                     TS.1        TS.2
## 2009-01-05  0.021924601  0.16479626
## 2009-01-04 -0.904325473 -0.07551388
## 2009-01-03  0.413237769 -2.33069954
## 2009-01-02  0.186621223 -0.52321193
## 2009-01-01  0.230817687 -1.51775046
## 2008-12-31  0.235680176  0.79178150
## 2008-12-30  1.483738895  0.57023860
## 2008-12-29  1.099462672  2.50495921
## 2008-12-28 -0.004685482 -0.25997610
## 2008-12-27 -1.421197846  0.20158998
## 2008-12-26 -0.132607774  0.47362237
```

timeSeries 函数同样能处理 R 中的 Date 型变量。

2.1.1.9 基于 POSIXct 标识创建时间序列对象时有哪些要注意的? 常规数据

```
set.seed(1953)
data <- rnorm(6)
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
z1 <- zoo(data, as.POSIXct(charvec))
z2 <- zoo(data, ISOdatetime(2009, 1:6, 1, 0, 0, 0))
```

```
z3 <- zoo(data, ISOdate(2009, 1:6, 1, 0))
z1
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.0219246 -0.9043255  0.4132378  0.1866212
## 2009-05-01 2009-06-01
##  0.2308177  0.2356802
```

```
z2
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.0219246 -0.9043255  0.4132378  0.1866212
## 2009-05-01 2009-06-01
##  0.2308177  0.2356802
```

```
z3
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.0219246 -0.9043255  0.4132378  0.1866212
## 2009-05-01 2009-06-01
##  0.2308177  0.2356802
```

注意，上面三种方式创建的 zoo 类型时间序列对象完全一致，无法从输出结果上判断出 zoo 对象时间戳是基于何种方式创建的。

xts:

```
x1 <- xts(data, as.POSIXct(charvec))
x2 <- xts(data, ISOdatetime(2009, 1:6, 1, 0, 0, 0))
x3 <- xts(data, ISOdate(2009, 1:6, 1, 0))
x1
```

```
##                   [,1]
## 2009-01-01  0.0219246
```

```
## 2009-02-01 -0.9043255
## 2009-03-01  0.4132378
## 2009-04-01  0.1866212
## 2009-05-01  0.2308177
## 2009-06-01  0.2356802
```

```
x2
```

```
##                  [,1]
## 2009-01-01  0.0219246
## 2009-02-01 -0.9043255
## 2009-03-01  0.4132378
## 2009-04-01  0.1866212
## 2009-05-01  0.2308177
## 2009-06-01  0.2356802
```

```
x3
```

```
## Warning: object timezone ('GMT') is different from system timezone ('')
##   NOTE: set 'options(xts_check_TZ = FALSE)' to disable this warning
##     This note is displayed once per session
```

```
##                  [,1]
## 2009-01-01  0.0219246
## 2009-02-01 -0.9043255
## 2009-03-01  0.4132378
## 2009-04-01  0.1866212
## 2009-05-01  0.2308177
## 2009-06-01  0.2356802
```

timeSeries:

```
s1 <- timeSeries(data, charvec)
s2 <- timeSeries(data, ISOdatetime(2009, 1:6, 1, 0, 0, 0))
s3 <- timeSeries(data, ISOdate(2009, 1:6, 1, 0))
s1
```

```
## GMT
##                   TS.1
## 2009-01-01  0.0219246
## 2009-02-01 -0.9043255
## 2009-03-01  0.4132378
## 2009-04-01  0.1866212
## 2009-05-01  0.2308177
## 2009-06-01  0.2356802
```

```
s2
```

```
## GMT
##                            TS.1
## 2008-12-31 16:00:00  0.0219246
## 2009-01-31 16:00:00 -0.9043255
## 2009-02-28 16:00:00  0.4132378
## 2009-03-31 16:00:00  0.1866212
## 2009-04-30 16:00:00  0.2308177
## 2009-05-31 16:00:00  0.2356802
```

```
s3
```

```
## GMT
##                   TS.1
## 2009-01-01  0.0219246
## 2009-02-01 -0.9043255
## 2009-03-01  0.4132378
## 2009-04-01  0.1866212
## 2009-05-01  0.2308177
## 2009-06-01  0.2356802
```

2.1.1.10 时间序列对象的标识/时间跟对象的创建方式有关吗? 常规数据

延用上例数据。

zoo:

```
class(index(z1))
```

```
## [1] "POSIXct" "POSIXt"
```

```
class(index(z2))
```

```
## [1] "POSIXct" "POSIXt"
```

```
class(index(z3))
```

```
## [1] "POSIXct" "POSIXt"
```

xts:

```
class(index(x1))
```

```
## [1] "POSIXct" "POSIXt"
```

```
class(index(x2))
```

```
## [1] "POSIXct" "POSIXt"
```

```
class(index(x3))
```

```
## [1] "POSIXct" "POSIXt"
```

timeSeries:

```
class(time(s1))
```

```
## [1] "timeDate"
## attr(,"package")
## [1] "timeDate"
```

```
class(time(s2))
```

```
## [1] "timeDate"
## attr(,"package")
## [1] "timeDate"
```

```
class(time(s3))
```

```
## [1] "timeDate"
## attr(,"package")
## [1] "timeDate"
```

2.1.2 规则时间序列对象

需要加载的 R 包。

```
library(zoo)
library(xts)
library(timeSeries)
```

首先，弄清楚什么是规则时间序列。oracle 数据库操作手册对此的定义如下：根据时间序列的新记录能否被预测可以讲时间序列分为规则时间序列和不规则时间序列。

规则时间序列，规则时间序列数据的新记录通常在规定的时间间隔内出现。比如，股票市场的日数据可以形成一个规则时间序列，股票 XYZ 自 1997 年以来的交易量序列、开盘价序列、最高价序列、最低价序列和收盘价序列可以构成规则时间序列数据。

不规则时间序列: 不规则时间序列数据的新纪录通常不在预计时刻出现，或者说不规则时间序列的样本点的时间戳不遵循特定的循环模型呢个。例如，ATM 及上的存入款和取出款数据构成一个不规则时间序列。不规则时间序列，经常会在某个时间段，没有数据记录，而在某个较短的时间段涌现出大量数据记录。

这里关于规则时间序列的定义跟 R 用户或者程序员脑海中的概念是不同的，在 R 中，时间戳标记（通常是基于日历日期）为等间隔的时间序列就是规则时间序列。这个概念更为严格。

例如：应用最广泛的规则时间序列是没有日时间戳和时刻时间戳标识的月度时间序列或者季度时间序列。在 R 中，两者是根据其与年的换算关系来进行描述，譬如，月度数据对应的参数是 12，因为一年有 12 个月，季度数据对应的参数是 4，因为一年有四季度。

```
args(ts)
```

```
## function (data = NA, start = 1, end = numeric(), frequency = 1,
##     deltat = 1, ts.eps = getOption("ts.eps"), class = if (nseries >
##         1) c("mts", "ts", "matrix", "array") else "ts", names = if (!is.null(dimname
##         seq(nseries)))
## NULL
```

```
data <- round(rnorm(24), 4)
ts(data, start = c(2008, 3), frequency = 12)
```

```
##          Jan     Feb     Mar     Apr     May
## 2008                  1.4837  1.0995 -0.0047
## 2009  0.7918  0.5702  2.5050 -0.2600  0.2016
## 2010 -1.6032  0.6542
##          Jun     Jul     Aug     Sep     Oct
## 2008 -1.4212 -0.1326  0.1648 -0.0755 -2.3307
## 2009  0.4736  0.4941 -0.3688  0.2365  0.6201
## 2010
##          Nov     Dec
```

```
## 2008 -0.5232 -1.5178
## 2009  0.2098 -1.4904
## 2010
```

```
ts(data, start = c(2008, 3), frequency = 4)
```

```
##         Qtr1    Qtr2    Qtr3    Qtr4
## 2008                  1.4837  1.0995
## 2009 -0.0047 -1.4212 -0.1326  0.1648
## 2010 -0.0755 -2.3307 -0.5232 -1.5178
## 2011  0.7918  0.5702  2.5050 -0.2600
## 2012  0.2016  0.4736  0.4941 -0.3688
## 2013  0.2365  0.6201  0.2098 -1.4904
## 2014 -1.6032  0.6542
```

2.1.2.1 如何创建一个规则的月度时间序列对象? 常规数据

```
data <- round(rnorm(24), 4)
```

ts:

```
tm <- ts(data, start = c(2008, 3), frequency = 12)
tm
```

```
##          Jan     Feb     Mar     Apr     May
## 2008                -0.1091 -0.6485  0.7964
## 2009  0.3347  0.3607 -1.3589 -0.4298 -0.3458
## 2010 -2.5492 -0.8575
##          Jun     Jul     Aug     Sep     Oct
## 2008  0.3177  0.2305 -1.5229 -0.1986 -2.0670
## 2009  1.0028 -0.2183  1.1060 -0.0444  0.4914
## 2010
##          Nov     Dec
## 2008  0.2459 -1.0007
```

```
## 2009 -1.4419  2.0744
## 2010
```

zoo:

```
zm <- zooreg(data, start = c(2008, 3), frequency = 12)
zm
```

```
##  3 2008  4 2008  5 2008  6 2008  7 2008  8 2008
## -0.1091 -0.6485  0.7964  0.3177  0.2305 -1.5229
##  9 2008 10 2008 11 2008 12 2008  1 2009  2 2009
## -0.1986 -2.0670  0.2459 -1.0007  0.3347  0.3607
##  3 2009  4 2009  5 2009  6 2009  7 2009  8 2009
## -1.3589 -0.4298 -0.3458  1.0028 -0.2183  1.1060
##  9 2009 10 2009 11 2009 12 2009  1 2010  2 2010
## -0.0444  0.4914 -1.4419  2.0744 -2.5492 -0.8575
```

xts:

```
xm <- as.xts(tm)
xm
```

```
##            [,1]
##  3 2008 -0.1091
##  4 2008 -0.6485
##  5 2008  0.7964
##  6 2008  0.3177
##  7 2008  0.2305
##  8 2008 -1.5229
##  9 2008 -0.1986
## 10 2008 -2.0670
## 11 2008  0.2459
## 12 2008 -1.0007
##  1 2009  0.3347
```

```
##  2 2009  0.3607
##  3 2009 -1.3589
##  4 2009 -0.4298
##  5 2009 -0.3458
##  6 2009  1.0028
##  7 2009 -0.2183
##  8 2009  1.1060
##  9 2009 -0.0444
## 10 2009  0.4914
## 11 2009 -1.4419
## 12 2009  2.0744
##  1 2010 -2.5492
##  2 2010 -0.8575
```

timeSeries:

```
sm <- as.timeSeries(tm)
sm
```

```
## GMT
##                TS.1
## 2008-03-31 -0.1091
## 2008-04-30 -0.6485
## 2008-05-31  0.7964
## 2008-06-30  0.3177
## 2008-07-31  0.2305
## 2008-08-31 -1.5229
## 2008-09-30 -0.1986
## 2008-10-31 -2.0670
## 2008-11-30  0.2459
## 2008-12-31 -1.0007
## 2009-01-31  0.3347
## 2009-02-28  0.3607
## 2009-03-31 -1.3589
```

```
## 2009-04-30 -0.4298
## 2009-05-31 -0.3458
## 2009-06-30  1.0028
## 2009-07-31 -0.2183
## 2009-08-31  1.1060
## 2009-09-30 -0.0444
## 2009-10-31  0.4914
## 2009-11-30 -1.4419
## 2009-12-31  2.0744
## 2010-01-31 -2.5492
## 2010-02-28 -0.8575
```

2.1.2.2 timeSeries 对象能以规则时间序列样式显示吗? 是的。

```
print(sm, style = "h")
```

```
## 2008-03-31 2008-04-30 2008-05-31 2008-06-30
##    -0.1091    -0.6485     0.7964     0.3177
## 2008-07-31 2008-08-31 2008-09-30 2008-10-31
##     0.2305    -1.5229    -0.1986    -2.0670
## 2008-11-30 2008-12-31 2009-01-31 2009-02-28
##     0.2459    -1.0007     0.3347     0.3607
## 2009-03-31 2009-04-30 2009-05-31 2009-06-30
##    -1.3589    -0.4298    -0.3458     1.0028
## 2009-07-31 2009-08-31 2009-09-30 2009-10-31
##    -0.2183     1.1060    -0.0444     0.4914
## 2009-11-30 2009-12-31 2010-01-31 2010-02-28
##    -1.4419     2.0744    -2.5492    -0.8575
```

```
print(sm, style = "h", format = "%Y %b")
```

2.1.2.3 能自定义 timeSeries 对象的日期格式吗?

```
## 2008  3 2008  4 2008  5 2008  6 2008  7 2008  8
## -0.1091 -0.6485  0.7964  0.3177  0.2305 -1.5229
## 2008  9 2008 10 2008 11 2008 12 2009  1 2009  2
## -0.1986 -2.0670  0.2459 -1.0007  0.3347  0.3607
## 2009  3 2009  4 2009  5 2009  6 2009  7 2009  8
## -1.3589 -0.4298 -0.3458  1.0028 -0.2183  1.1060
## 2009  9 2009 10 2009 11 2009 12 2010  1 2010  2
## -0.0444  0.4914 -1.4419  2.0744 -2.5492 -0.8575
```

```
print(sm, style = "h", format = "%Y(%m)")
```

```
## 2008(03) 2008(04) 2008(05) 2008(06) 2008(07)
##  -0.1091  -0.6485   0.7964   0.3177   0.2305
## 2008(08) 2008(09) 2008(10) 2008(11) 2008(12)
##  -1.5229  -0.1986  -2.0670   0.2459  -1.0007
## 2009(01) 2009(02) 2009(03) 2009(04) 2009(05)
##   0.3347   0.3607  -1.3589  -0.4298  -0.3458
## 2009(06) 2009(07) 2009(08) 2009(09) 2009(10)
##   1.0028  -0.2183   1.1060  -0.0444   0.4914
## 2009(11) 2009(12) 2010(01) 2010(02)
##  -1.4419   2.0744  -2.5492  -0.8575
```

2.1.2.4 如何创建一个规则的季度时间序列对象? 常规数据

```
data <- round(rnorm(24), 4)
```

ts:

```
tq <- ts(data, start = c(2008, 3), frequency = 4)
tq
```

```
##         Qtr1    Qtr2    Qtr3    Qtr4
## 2008                  0.4469 -0.2412
## 2009 -0.3665  1.2000  0.5114  0.4695
```

```
## 2010  0.2093  1.2756 -2.1948 -0.7763
## 2011  0.3733 -0.6908  1.1456 -1.5655
## 2012 -1.5795  0.2059  0.5029 -0.4230
## 2013  0.6698  0.5941 -1.6126 -0.4459
## 2014  0.4151 -1.0811
```

zoo:

```
zq <- zooreg(data, start = c(2008, 3), frequency = 4)
zq
```

```
## 2008 Q3 2008 Q4 2009 Q1 2009 Q2 2009 Q3 2009 Q4
##  0.4469 -0.2412 -0.3665  1.2000  0.5114  0.4695
## 2010 Q1 2010 Q2 2010 Q3 2010 Q4 2011 Q1 2011 Q2
##  0.2093  1.2756 -2.1948 -0.7763  0.3733 -0.6908
## 2011 Q3 2011 Q4 2012 Q1 2012 Q2 2012 Q3 2012 Q4
##  1.1456 -1.5655 -1.5795  0.2059  0.5029 -0.4230
## 2013 Q1 2013 Q2 2013 Q3 2013 Q4 2014 Q1 2014 Q2
##  0.6698  0.5941 -1.6126 -0.4459  0.4151 -1.0811
```

xts:

```
xq <- as.xts(tq)
head(xq)
```

```
##            [,1]
## 2008 Q3  0.4469
## 2008 Q4 -0.2412
## 2009 Q1 -0.3665
## 2009 Q2  1.2000
## 2009 Q3  0.5114
## 2009 Q4  0.4695
```

timeSeries:

```
sq <- as.timeSeries(tq)
head(sq)
```

```
## GMT
##                TS.1
## 2008-09-30  0.4469
## 2008-12-31 -0.2412
## 2009-03-31 -0.3665
## 2009-06-30  1.2000
## 2009-09-30  0.5114
## 2009-12-31  0.4695
```

2.1.2.5 timeSeries 对象能以规则时间序列的样式显示吗? 是的。

```
print(sq, style = "h")
```

```
## 2008-09-30 2008-12-31 2009-03-31 2009-06-30
##     0.4469    -0.2412    -0.3665     1.2000
## 2009-09-30 2009-12-31 2010-03-31 2010-06-30
##     0.5114     0.4695     0.2093     1.2756
## 2010-09-30 2010-12-31 2011-03-31 2011-06-30
##    -2.1948    -0.7763     0.3733    -0.6908
## 2011-09-30 2011-12-31 2012-03-31 2012-06-30
##     1.1456    -1.5655    -1.5795     0.2059
## 2012-09-30 2012-12-31 2013-03-31 2013-06-30
##     0.5029    -0.4230     0.6698     0.5941
## 2013-09-30 2013-12-31 2014-03-31 2014-06-30
##    -1.6126    -0.4459     0.4151    -1.0811
```

2.1.2.6 能自定义 timeSeries 对象的时间格式吗? 可以!

```
print(sq, style = "h", format = "%Y %b")
```

```
## 2008  9 2008 12 2009  3 2009  6 2009  9 2009 12
##  0.4469 -0.2412 -0.3665  1.2000  0.5114  0.4695
## 2010  3 2010  6 2010  9 2010 12 2011  3 2011  6
##  0.2093  1.2756 -2.1948 -0.7763  0.3733 -0.6908
## 2011  9 2011 12 2012  3 2012  6 2012  9 2012 12
##  1.1456 -1.5655 -1.5795  0.2059  0.5029 -0.4230
## 2013  3 2013  6 2013  9 2013 12 2014  3 2014  6
##  0.6698  0.5941 -1.6126 -0.4459  0.4151 -1.0811
```

```
print(sq, style = "h", format = "%Q")
```

```
## 2008 Q3 2008 Q4 2009 Q1 2009 Q2 2009 Q3 2009 Q4
##  0.4469 -0.2412 -0.3665  1.2000  0.5114  0.4695
## 2010 Q1 2010 Q2 2010 Q3 2010 Q4 2011 Q1 2011 Q2
##  0.2093  1.2756 -2.1948 -0.7763  0.3733 -0.6908
## 2011 Q3 2011 Q4 2012 Q1 2012 Q2 2012 Q3 2012 Q4
##  1.1456 -1.5655 -1.5795  0.2059  0.5029 -0.4230
## 2013 Q1 2013 Q2 2013 Q3 2013 Q4 2014 Q1 2014 Q2
##  0.6698  0.5941 -1.6126 -0.4459  0.4151 -1.0811
```

2.1.2.7 如何知道一个时间序列对象是否是规则时间序列呢? zoo:

```
z <- zooreg(data, start = c(2008, 3), frequency = 4)
is.regular(z)
```

```
## [1] TRUE
```

2.1.3 时区和夏令时

需要载入的 R 包。

```
library(zoo)
library(xts)
library(timeSeries)
```

2.1.3.1 如何创建一个与时区相关的时间序列对象? 创建一个苏黎世所在的中欧时区的时间序列。

常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
z1.zrh <- zoo(data, as.POSIXct(charvec, tz = "CET"))
z1.zrh
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          4
## 2009-05-01 2009-06-01
##          5          6
```

xts:

```
x1.zrh <- xts(data, as.POSIXct(charvec, tz = "CET"))
x1.zrh
```

```
## Warning: object timezone ('CET') is different
## from system timezone ('')
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

最先版本的 xts 包已经不再支持上述功能，现在 xts 会强制的将时间戳对应到 GMT 时区。

timeSeries:

```
s1.zrh <- timeSeries(data, charvec, zone = "Zurich", FinCenter = "Zurich")
s1.zrh
```

```
## Zurich
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

timeSeries 函数有两个与时区设置相关的参数。第一个参数 zone，用以设定创建时间序列对象是所用到的时间戳标识所属的时区信息和金融中心信息；第二个参数是 FinCenter, 用以设定将来显示或者调用时间序列对象时所在的时区或者金融中心。

```
args(timeSeries)
```

```
## function (data, charvec, units = NULL, format = NULL, zone = "",
##     FinCenter = "", recordIDs = data.frame(), title = NULL, documentation = NULL,
##     ...)
## NULL
```

查看一下，因为参数设置不同所能产生的四种情况。

```
timeSeries(data, charvec, zone = "Zurich", FinCenter = "Zurich",
    units = "s1.zrh.zrh")
```

```
## Zurich
##            s1.zrh.zrh
## 2009-01-01          1
## 2009-02-01          2
## 2009-03-01          3
## 2009-04-01          4
## 2009-05-01          5
## 2009-06-01          6
```

```
timeSeries(data, charvec, zone = "GMT", FinCenter = "Zurich",
    units = "s1.gmt.zrh")
```

```
## Zurich
##                     s1.gmt.zrh
## 2009-01-01 01:00:00          1
## 2009-02-01 01:00:00          2
## 2009-03-01 01:00:00          3
## 2009-04-01 02:00:00          4
## 2009-05-01 02:00:00          5
## 2009-06-01 02:00:00          6
```

```
timeSeries(data, charvec, zone = "Zurich", FinCenter = "GMT",
    units = "s1.zrh.gmt")
```

```
## GMT
##                     s1.zrh.gmt
## 2008-12-31 23:00:00          1
## 2009-01-31 23:00:00          2
## 2009-02-28 23:00:00          3
```

```
## 2009-03-31 22:00:00          4
## 2009-04-30 22:00:00          5
## 2009-05-31 22:00:00          6
```

```
timeSeries(data, charvec, zone = "GMT", FinCenter = "GMT", units = "s1.gmt.gmt")
```

```
## GMT
##             s1.gmt.gmt
## 2009-01-01          1
## 2009-02-01          2
## 2009-03-01          3
## 2009-04-01          4
## 2009-05-01          5
## 2009-06-01          6
```

2.1.3.2 查看时间序列时，能够看到时间序列的时区信息? 常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

注意：zoo 对象的时间戳必须是支持时区设置的时间戳类型、比如 POSIX 类型。

```
z1.zrh <- zoo(data, as.POSIXct(charvec, tz = "CET"))
z1.zrh
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          4
## 2009-05-01 2009-06-01
##          5          6
```

zoo 时间序列对象在显示时，不会自动显示时区信息。

xts:

xts 对象的时间戳必须是支持时区设置的时间戳类型、比如 POSIX 类型。

```
x1.zrh <- xts(data, as.POSIXct(charvec, tz = "CET"))
x1.zrh
```

```
## Warning: object timezone ('CET') is different
## from system timezone ('')
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

timeSeries:

```
s1.zrh <- timeSeries(data, charvec, zone = "Zurich", FinCenter = "Zurich")
s1.zrh
```

```
## Zurich
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

注意：timeSeries 对象在显示会在头部显示对象所属的时区信息（或者金融中心信息）。

该信息提取自 timeDate 类型的时间戳对象。

2.1.3.3 如何查看时间序列所属的时区信息? 对于 zoo 对象和 xts 对象，可以用 index 函数来查看时区信息；对于 timeSeries 对象，用 time 函数。

常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
z1.zrh <- zoo(data, as.POSIXct(charvec, tz = "CET"))
index(z1.zrh)
```

```
## [1] "2009-01-01 CET"  "2009-02-01 CET"
## [3] "2009-03-01 CET"  "2009-04-01 CEST"
## [5] "2009-05-01 CEST" "2009-06-01 CEST"
```

xts:

```
x1.zrh <- xts(data, as.POSIXct(charvec, tz = "CET"))
index(x1.zrh)
```

```
## [1] "2009-01-01 CET"  "2009-02-01 CET"
## [3] "2009-03-01 CET"  "2009-04-01 CEST"
## [5] "2009-05-01 CEST" "2009-06-01 CEST"
```

timeSeries:

```
s1.zrh <- timeSeries(data, charvec, zone = "Zurich", FinCenter = "Zurich")
time(s1.zrh)
```

```
## Zurich
## [1] [2009-01-01] [2009-02-01] [2009-03-01]
## [4] [2009-04-01] [2009-05-01] [2009-06-01]
```

注意：也可以用 finCenter() 函数来保留或者设置 timeSeries 对象的时区信息。

```
currentCenter <- finCenter(s1.zrh)
currentCenter
```

```
## [1] "Zurich"
```

2.1.3.4 如何查看创建时间序列对象时用到的夏令时规则呢? zoo 及 xts 对象:

目前无法做到。

timeSeries:

对于 timeSeries 对象，可以通过查看夏令时规则的数据框获知创建时间序列对象时的时区信息。为了简便，下面展示部分夏令时规则的记录数据:

```
Zurich()[54:64, ]
```

```
##                 Zurich offSet isdst TimeZone
## 54 2004-10-31 01:00:00   3600     0      CET
## 55 2005-03-27 01:00:00   7200     1     CEST
## 56 2005-10-30 01:00:00   3600     0      CET
## 57 2006-03-26 01:00:00   7200     1     CEST
## 58 2006-10-29 01:00:00   3600     0      CET
## 59 2007-03-25 01:00:00   7200     1     CEST
## 60 2007-10-28 01:00:00   3600     0      CET
```

```
## 61 2008-03-30 01:00:00   7200     1     CEST
## 62 2008-10-26 01:00:00   3600     0      CET
## 63 2009-03-29 01:00:00   7200     1     CEST
## 64 2009-10-25 01:00:00   3600     0      CET
##       numeric
## 54 1099184400
## 55 1111885200
## 56 1130634000
## 57 1143334800
## 58 1162083600
## 59 1174784400
## 60 1193533200
## 61 1206838800
## 62 1224982800
## 63 1238288400
## 64 1256432400
```

上表展示苏黎世的夏令时变化规律。第一列为苏黎世当地时间，第二列为苏黎世时间与 GMT 时间的延迟（以秒计），第三列标识夏令时规则是否在执行，第四列为时区缩写，第五列标识时间戳变化时的秒针信息。

2.1.3.5 创建时间序列对象时，能用哪些时区呢? zoo 和 xts:

不知道 zoo 和 xts 包具体支持哪些时区。

2.1.3.6 你能告诉我 Mumbai 股票交易所位于按个时区吗? 这个问题没有固定答案。Unix 系统下，可以通过查看/usr/share/tzone 找到 zoo 和 xts 包找到支持的区列表。

timeSeries:

对于 timeSeries 对象，可用 listFinCenter 函数来查看所有被支持的金融中心的时区列表。

```
length(listFinCenter())
```

```
## [1] 311
```

结果太多，无法全部显示。选取太平洋地区的部分时区和首字母为 L 的全部时区：

```
listFinCenter("Pacific")
```

```
##  [1] "Pacific/Apia"
##  [2] "Pacific/Auckland"
##  [3] "Pacific/Bougainville"
##  [4] "Pacific/Chatham"
##  [5] "Pacific/Easter_Island"
##  [6] "Pacific/Efate"
##  [7] "Pacific/Fakaofo"
##  [8] "Pacific/Fiji"
##  [9] "Pacific/Galapagos"
## [10] "Pacific/Gambier"
## [11] "Pacific/Guadalcanal"
## [12] "Pacific/Guam"
## [13] "Pacific/Honolulu"
## [14] "Pacific/Kanton"
## [15] "Pacific/Kiritimati"
## [16] "Pacific/Kosrae"
## [17] "Pacific/Kwajalein"
## [18] "Pacific/Marquesas"
## [19] "Pacific/Nauru"
## [20] "Pacific/Niue"
## [21] "Pacific/Norfolk"
## [22] "Pacific/Noumea"
## [23] "Pacific/Pago_Pago"
## [24] "Pacific/Palau"
```

```
## [25] "Pacific/Pitcairn"
## [26] "Pacific/Port_Moresby"
## [27] "Pacific/Rarotonga"
## [28] "Pacific/Tahiti"
## [29] "Pacific/Tarawa"
## [30] "Pacific/Tongatapu"
```

```
listFinCenter(".*/L")
```

```
##  [1] "Africa/Lagos"
##  [2] "America/Argentina/La_Rioja"
##  [3] "America/Kentucky/Louisville"
##  [4] "America/La_Paz"
##  [5] "America/Lima"
##  [6] "America/Los_Angeles"
##  [7] "Australia/Lindeman"
##  [8] "Australia/Lord_Howe"
##  [9] "Europe/Lisbon"
## [10] "Europe/London"
```

甚至可以定制自己的金融中心。譬如，德国的夏令时规则为 “Germany/Berlin”，身为银行家的你却更偏好用法兰克福时区，你可以定制自己的时区列表。

```
Frankfurt <- Berlin
timeSeries(runif(1:12), timeCalendar(), zone = "Frankfurt", FinCenter = "Frankfurt")
```

```
## Frankfurt
##                   TS.1
## 2025-01-01 0.30640419
## 2025-02-01 0.93921809
## 2025-03-01 0.91220179
## 2025-04-01 0.68942634
## 2025-05-01 0.68014228
```

```
## 2025-06-01 0.85762041
## 2025-07-01 0.09803107
## 2025-08-01 0.62674869
## 2025-09-01 0.20364073
## 2025-10-01 0.30498536
## 2025-11-01 0.02438005
## 2025-12-01 0.08416704
```

注意，timeCalendar 函数来自于 timeDate 包中，其作用是用来当前年份创建月度时间戳。

2.1.3.7 如何改变现有时间序列对象的时区信息？ zoo 及 xts 对象:

没有直接方法。建议先提取出时间戳，将新旧时区的差值加上之后，用新的时间戳覆盖旧时间戳。

timeSeries:

对于 timeSeries 对象，可以用 finCenter 函数来改变现有数据的时区信息。譬如，有一个在伦敦记录的时间序列数据，现在要调整为纽约当地时间，以备在苏黎世查看。

```
ZRH <- timeSeries(rnorm(6), timeCalendar(2009)[7:12], zone = "London",
    FinCenter = "Zurich")

ZRH
```

```
## Zurich
##                            TS.1
## 2009-07-01 01:00:00 -0.39767135
## 2009-08-01 01:00:00 -0.02014968
## 2009-09-01 01:00:00  0.88326587
## 2009-10-01 01:00:00  0.14669883
## 2009-11-01 01:00:00  0.46105243
## 2009-12-01 01:00:00 -0.41846085
```

```
finCenter(ZRH) <- "New_York"
ZRH
```

```
## New_York
##                             TS.1
## 2009-06-30 19:00:00 -0.39767135
## 2009-07-31 19:00:00 -0.02014968
## 2009-08-31 19:00:00  0.88326587
## 2009-09-30 19:00:00  0.14669883
## 2009-10-31 20:00:00  0.46105243
## 2009-11-30 19:00:00 -0.41846085
```

上例反映出，在欧洲和美国，夏令时与冬令时的转变月份不同，准确的说，一个在 10 月，一个在 11 月。查看一下夏令时表确认一下。

```
Zurich()[63, ]
```

```
##                  Zurich offSet isdst TimeZone
## 63 2009-03-29 01:00:00   7200     1     CEST
##       numeric
## 63 1238288400
```

```
New_York()[179, ]
```

```
##                New_York offSet isdst TimeZone
## 179 2009-11-01 06:00:00 -18000     0      EST
##        numeric
## 179 1257055200
```

2.1.3.8 数据记录的两个城市，属于同一个时区但有不同的 DST 规则，如何处理这样的情况呢? zoo 及 xts 对象:

无法处理。

timeSeries:

1940 至 1985 年间，德国和瑞士发生过多次类似情况。来看一下包含 DST 规则表：

```
Berlin()[8:38, ]
```

```
##                  Berlin offSet isdst TimeZone
## 8  1940-04-01 01:00:00   7200     1     CEST
## 9  1942-11-02 01:00:00   3600     0      CET
## 10 1943-03-29 01:00:00   7200     1     CEST
## 11 1943-10-04 01:00:00   3600     0      CET
## 12 1944-04-03 01:00:00   7200     1     CEST
## 13 1944-10-02 01:00:00   3600     0      CET
## 14 1945-04-02 01:00:00   7200     1     CEST
## 15 1945-05-24 00:00:00  10800     1     CEMT
## 16 1945-09-24 00:00:00   7200     1     CEST
## 17 1945-11-18 01:00:00   3600     0      CET
## 18 1946-04-14 01:00:00   7200     1     CEST
## 19 1946-10-07 01:00:00   3600     0      CET
## 20 1947-04-06 02:00:00   7200     1     CEST
## 21 1947-05-11 01:00:00  10800     1     CEMT
## 22 1947-06-29 00:00:00   7200     1     CEST
## 23 1947-10-05 01:00:00   3600     0      CET
## 24 1948-04-18 01:00:00   7200     1     CEST
## 25 1948-10-03 01:00:00   3600     0      CET
## 26 1949-04-10 01:00:00   7200     1     CEST
## 27 1949-10-02 01:00:00   3600     0      CET
## 28 1980-04-06 01:00:00   7200     1     CEST
## 29 1980-09-28 01:00:00   3600     0      CET
## 30 1981-03-29 01:00:00   7200     1     CEST
## 31 1981-09-27 01:00:00   3600     0      CET
## 32 1982-03-28 01:00:00   7200     1     CEST
## 33 1982-09-26 01:00:00   3600     0      CET
## 34 1983-03-27 01:00:00   7200     1     CEST
## 35 1983-09-25 01:00:00   3600     0      CET
```

```
## 36 1984-03-25 01:00:00   7200     1     CEST
## 37 1984-09-30 01:00:00   3600     0      CET
## 38 1985-03-31 01:00:00   7200     1     CEST
##       numeric
## 8  -938905200
## 9  -857257200
## 10 -844556400
## 11 -828226800
## 12 -812502000
## 13 -796777200
## 14 -781052400
## 15 -776563200
## 16 -765936000
## 17 -761180400
## 18 -748479600
## 19 -733273200
## 20 -717631200
## 21 -714610800
## 22 -710380800
## 23 -701910000
## 24 -684975600
## 25 -670460400
## 26 -654130800
## 27 -639010800
## 28  323830800
## 29  338950800
## 30  354675600
## 31  370400400
## 32  386125200
## 33  401850000
## 34  417574800
## 35  433299600
## 36  449024400
```

```
## 37  465354000
## 38  481078800
```

```
Zurich()[2:15, ]
```

```
##                  Zurich offSet isdst TimeZone
## 2  1894-05-31 23:30:14   3600     0      CET
## 3  1941-05-05 00:00:00   7200     1     CEST
## 4  1941-10-06 00:00:00   3600     0      CET
## 5  1942-05-04 00:00:00   7200     1     CEST
## 6  1942-10-05 00:00:00   3600     0      CET
## 7  1981-03-29 01:00:00   7200     1     CEST
## 8  1981-09-27 01:00:00   3600     0      CET
## 9  1982-03-28 01:00:00   7200     1     CEST
## 10 1982-09-26 01:00:00   3600     0      CET
## 11 1983-03-27 01:00:00   7200     1     CEST
## 12 1983-09-25 01:00:00   3600     0      CET
## 13 1984-03-25 01:00:00   7200     1     CEST
## 14 1984-09-30 01:00:00   3600     0      CET
## 15 1985-03-31 01:00:00   7200     1     CEST
##        numeric
## 2  -2385246586
## 3   -904435200
## 4   -891129600
## 5   -872985600
## 6   -859680000
## 7    354675600
## 8    370400400
## 9    386125200
## 10   401850000
## 11   417574800
## 12   433299600
## 13   449024400
## 14   465354000
```

```
## 15   481078800
```

在苏黎世当地时间记录的入夜时间为 16:00，假如我们想在柏林应用这一批数据。问题是，最早能在柏林当地时间的什么时候，来开始调查这批数据呢?

```
charvec <- paste("1980-0", 2:5, "-15 16:00:00", sep = "")
charvec
```

```
## [1] "1980-02-15 16:00:00" "1980-03-15 16:00:00"
## [3] "1980-04-15 16:00:00" "1980-05-15 16:00:00"
```

```
timeSeries(runif(4), charvec, zone = "Zurich", FinCenter = "Berlin",
    units = "fromZurich")
```

```
## Berlin
##                     fromZurich
## 1980-02-15 16:00:00  0.7609298
## 1980-03-15 16:00:00  0.7077751
## 1980-04-15 17:00:00  0.5981918
## 1980-05-15 17:00:00  0.9540911
```

在 2 月和 3 月，我们能在伯林，用和苏黎世相同的时间 16:00，来开始我们的调查。而在 4 月和 5 月，我们却要在比苏黎世时间晚一个小时的 17:00 开始调查。

2.1.4 时间序列对象的排序以及重合日期处理

需要载入的 R 包

```
library(zoo)
library(xts)
library(timeSeries)
```

2.1.4.1 如何创建一个逆序的时间序列数据? 常规数据

```
set.seed <- 1953
data <- rnorm(6)
charvec <- rev(paste("2009-0", 1:6, "-01", sep = ""))
charvec
```

```
## [1] "2009-06-01" "2009-05-01" "2009-04-01"
## [4] "2009-03-01" "2009-02-01" "2009-01-01"
```

注意: 字符型向量 charvec 是逆序的。

zoo 及 xts 对象:

```
zoo(data, as.Date(charvec))
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
## -0.1785927  0.6680581  0.5480112 -0.1579903
## 2009-05-01 2009-06-01
## -1.2478298  0.6717223
```

```
xts(data, as.Date(charvec))
```

```
##                  [,1]
## 2009-01-01 -0.1785927
## 2009-02-01  0.6680581
## 2009-03-01  0.5480112
## 2009-04-01 -0.1579903
## 2009-05-01 -1.2478298
## 2009-06-01  0.6717223
```

zoo 对象和 xts 对象一般会被强制转化为正序，因此，无法创建逆序的时间序列对象。

timeSeries:

timeSeries 对象可以是正序的，也可以是逆序的，甚至于可以随机排序。

```
tS <- timeSeries(data, charvec)
tS
```

```
## GMT
##                   TS.1
## 2009-06-01  0.6717223
## 2009-05-01 -1.2478298
## 2009-04-01 -0.1579903
## 2009-03-01  0.5480112
## 2009-02-01  0.6680581
## 2009-01-01 -0.1785927
```

取逆序

```
rev(tS)
```

```
## GMT
##                   TS.1
## 2009-01-01 -0.1785927
## 2009-02-01  0.6680581
## 2009-03-01  0.5480112
## 2009-04-01 -0.1579903
## 2009-05-01 -1.2478298
## 2009-06-01  0.6717223
```

随机排序

```
sample(tS)
```

```
## GMT
##                   TS.1
## 2009-02-01  0.6680581
## 2009-06-01  0.6717223
## 2009-04-01 -0.1579903
```

```
## 2009-05-01 -1.2478298
## 2009-03-01  0.5480112
## 2009-01-01 -0.1785927
```

2.1.4.2 如何创建一个时期重叠的时间戳?

和创建非时期重叠的时间序列的方法相同。

常规数据

```
data1 <- c(1:6, 0)
charvec1 <- c(paste("2009-0", 1:6, "-01", sep = ""), "2009-04-01")
charvec1
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-04-01"
```

```
data2 <- 0:6
charvec2 <- c("2009-04-01", paste("2009-0", 1:6, "-01", sep = ""))
charvec2
```

```
## [1] "2009-04-01" "2009-01-01" "2009-02-01"
## [4] "2009-03-01" "2009-04-01" "2009-05-01"
## [7] "2009-06-01"
```

zoo:

```
zoo(data1, as.Date(charvec1))
```

```
## Warning in zoo(data1, as.Date(charvec1)): some
## methods for "zoo" objects do not work if the
## index entries in 'order.by' are not unique
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
```

```
##          1          2          3          4
## 2009-04-01 2009-05-01 2009-06-01
##          0          5          6
```

```
zoo(data2, as.Date(charvec2))
```

```
## Warning in zoo(data2, as.Date(charvec2)): some
## methods for "zoo" objects do not work if the
## index entries in 'order.by' are not unique

## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          0
## 2009-04-01 2009-05-01 2009-06-01
##          4          5          6
```

zoo() 将会返回一个警告信息，声明 zoo() 方法出现了错误。

xts:

xts 对象对 zoo 对象进行了扩展，其能支持时期重叠的时间戳。

```
xts(data1, as.Date(charvec1))
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-04-01    0
## 2009-05-01    5
## 2009-06-01    6
```

```
xts(data2, as.Date(charvec2))
```

```
##            [,1]
```

```
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    0
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

timeSeries:

timeSeries 对象支持时期重叠的时间戳。

```
timeSeries(data1, charvec1)
```

```
## GMT
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
## 2009-04-01    0
```

```
timeSeries(data2, charvec2)
```

```
## GMT
##            TS.1
## 2009-04-01    0
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

这种方法下创建的时间戳与 charvec 的时间戳顺序相同。这是 timeSeries 对象的特色，其可以记录用户记录时间序列时所使用的时间戳顺序。对 timeSeries 对象进行排序可以用 sort 函数。

```
sort(timeSeries(data1, charvec1))
```

```
## GMT
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-04-01    0
## 2009-05-01    5
## 2009-06-01    6
```

```
sort(timeSeries(data2, charvec2))
```

```
## GMT
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    0
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

也可以将 timeSeries 对象进行逆序排列。

```
sort(timeSeries(data1, charvec1), decreasing = TRUE)
```

```
## GMT
##            TS.1
```

```
## 2009-06-01    6
## 2009-05-01    5
## 2009-04-01    4
## 2009-04-01    0
## 2009-03-01    3
## 2009-02-01    2
## 2009-01-01    1
```

如果想保留数据的初始顺序记录，@recordIDs 序列中, 下面演示一下如何保存和提取这种信息。

```
args(timeSeries)
```

```
## function (data, charvec, units = NULL, format = NULL, zone = "",
##     FinCenter = "", recordIDs = data.frame(), title = NULL, documentation = NULL,
##     ...)
## NULL
```

```
data3 <- round(rnorm(7), 2)
charvec3 <- sample(charvec1)
tS <- sort(timeSeries(data3, charvec3, recordIDs = data.frame(1:7)))
tS
```

```
## GMT
##              TS.1 X1.7*
## 2009-01-01 -0.21     6
## 2009-02-01 -0.59     3
## 2009-03-01 -0.86     5
## 2009-04-01 -1.29     2
## 2009-04-01  1.23     4
## 2009-05-01 -1.42     1
## 2009-06-01 -0.25     7
```

现在用同样的方法来提取出排序的信息

```
tS@recordIDs
```

```
##   X1.7
## 6    6
## 3    3
## 5    5
## 2    2
## 4    4
## 1    1
## 7    7
```

也可以用直接对比性报告的形式将其打印出来。

```
cbind(series(tS), as.matrix(tS@recordIDs))
```

```
##             TS.1 X1.7
## 2009-01-01 -0.21    6
## 2009-02-01 -0.59    3
## 2009-03-01 -0.86    5
## 2009-04-01 -1.29    2
## 2009-04-01  1.23    4
## 2009-05-01 -1.42    1
## 2009-06-01 -0.25    7
```

```
data3
```

```
## [1] -1.42 -1.29 -0.59  1.23 -0.86 -0.21 -0.25
```

2.1.4.3 如何处理时间序列对象的附加属性信息? 考虑一个稍微复杂一些的例子。现在有来自不同公司的 dates, dateOfOffer, price offers, offered-Price 数据，以及公司名称和折扣率数据。供应商和折扣率信息被保存在一个名为 priceInfo 的数据框中。

```
offeredPrice <- 100 - c(3.4, 3.2, 4, 4, 4.1, 3.5, 2.9)
offeredPrice
```

```
## [1] 96.6 96.8 96.0 96.0 95.9 96.5 97.1
```

```
dateOfOffer <- paste("2009-0", c(1:3, 3, 4:6), "-01", sep = "")
dateOfOffer
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-03-01" "2009-04-01" "2009-05-01"
## [7] "2009-06-01"
```

```
providerCompany <- c(rep("UISA Ltd", times = 3), "HK Company",
    rep("UISA Ltd
", times = 3))
providerCompany
```

```
## [1] "UISA Ltd"    "UISA Ltd"    "UISA Ltd"
## [4] "HK Company"  "UISA Ltd \n" "UISA Ltd \n"
## [7] "UISA Ltd \n"
```

```
ratingOfOffer <- c(rep("AAA", times = 3), "BBB", rep("AAB", times = 3))
ratingOfOffer
```

```
## [1] "AAA" "AAA" "AAA" "BBB" "AAB" "AAB" "AAB"
```

```
priceInfo <- data.frame(providerCompany, ratingOfOffer)
priceInfo
```

```
##   providerCompany ratingOfOffer
## 1        UISA Ltd           AAA
## 2        UISA Ltd           AAA
## 3        UISA Ltd           AAA
```

```
## 4      HK Company           BBB
## 5     UISA Ltd \n           AAB
## 6     UISA Ltd \n           AAB
## 7     UISA Ltd \n           AAB
```

zoo:

基于日期和价格创建了一个 zoo 时间序列对象，同时将供应商和折扣率的信息保存在 info 属性中。

```
zp <- zoo(offeredPrice, as.Date(dateOfOffer))
```

```
## Warning in zoo(offeredPrice,
## as.Date(dateOfOffer)): some methods for "zoo"
## objects do not work if the index entries in
## 'order.by' are not unique
```

```
attr(zp, "info") = priceInfo
zp
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-03-01
##       96.6       96.8       96.0       96.0
## 2009-04-01 2009-05-01 2009-06-01
##       95.9       96.5       97.1
```

```
zp
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-03-01
##       96.6       96.8       96.0       96.0
## 2009-04-01 2009-05-01 2009-06-01
##       95.9       96.5       97.1
```

```
attr(zp, "info")
```

```
##   providerCompany ratingOfOffer
## 1        UISA Ltd           AAA
## 2        UISA Ltd           AAA
## 3        UISA Ltd           AAA
## 4      HK Company           BBB
## 5     UISA Ltd \n           AAB
## 6     UISA Ltd \n           AAB
## 7     UISA Ltd \n           AAB
```

xts:

与上面操作相同。

```
xp <- xts(offeredPrice, as.Date(dateOfOffer))
attr(xp, "info") = priceInfo
xp
```

```
##            [,1]
## 2009-01-01 96.6
## 2009-02-01 96.8
## 2009-03-01 96.0
## 2009-03-01 96.0
## 2009-04-01 95.9
## 2009-05-01 96.5
## 2009-06-01 97.1
```

```
attr(xp, "info")
```

```
##   providerCompany ratingOfOffer
## 1        UISA Ltd           AAA
## 2        UISA Ltd           AAA
## 3        UISA Ltd           AAA
## 4      HK Company           BBB
## 5     UISA Ltd \n           AAB
```

```
## 6     UISA Ltd \n           AAB
## 7     UISA Ltd \n           AAB
```

timeSeries:

```
tS <- timeSeries(offeredPrice, dateOfOffer, recordIDs = priceInfo)
tS
```

```
## GMT
##            TS.1 providerCompany* ratingOfOffer*
## 2009-01-01 96.6         UISA Ltd            AAA
## 2009-02-01 96.8         UISA Ltd            AAA
## 2009-03-01 96.0         UISA Ltd            AAA
## 2009-03-01 96.0       HK Company            BBB
## 2009-04-01 95.9      UISA Ltd \n            AAB
## 2009-05-01 96.5      UISA Ltd \n            AAB
## 2009-06-01 97.1      UISA Ltd \n            AAB
```

```
tS@recordIDs
```

```
##   providerCompany ratingOfOffer
## 1        UISA Ltd           AAA
## 2        UISA Ltd           AAA
## 3        UISA Ltd           AAA
## 4      HK Company           BBB
## 5     UISA Ltd \n           AAB
## 6     UISA Ltd \n           AAB
## 7     UISA Ltd \n           AAB
```

对于 timeSeries 对象而言， @recordsID 序列进行处理。

2.1.4.4 当调整时间序列时，对属性有什么影响? 考虑前面的一个例子。从上面的时间序列中移除第 4 个 offer 记录。

zoo:

```
zx <- zp[-4]
zx
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##       96.6       96.8       96.0       95.9
## 2009-05-01 2009-06-01
##       96.5       97.1
```

```
attr(zx, "info")
```

```
## NULL
```

经过这个操纵，zoo 对象的属性被删除了。只能重新核查并重新添加余下的属性信息。

```
zx
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##       96.6       96.8       96.0       95.9
## 2009-05-01 2009-06-01
##       96.5       97.1
```

```
attr(zx, "info") <- priceInfo[-4, ]
zx
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##       96.6       96.8       96.0       95.9
## 2009-05-01 2009-06-01
##       96.5       97.1
```

观察一下同样的操作对 xts 的影响。首先，删除了第四个观测，再把结果展示出来。

```
xx <- xp[-4]
xx
```

```
##            [,1]
## 2009-01-01 96.6
## 2009-02-01 96.8
## 2009-03-01 96.0
## 2009-04-01 95.9
## 2009-05-01 96.5
## 2009-06-01 97.1
```

与 zoo 不同，xts 没有删除全部属性信息，不过，其保留了全部属性信息。一个补救的办法是删除该记录的属性。

```
attr(xx, "info") <- attr(xx, "info")[-4, ]
xx
```

```
##            [,1]
## 2009-01-01 96.6
## 2009-02-01 96.8
## 2009-03-01 96.0
## 2009-04-01 95.9
## 2009-05-01 96.5
## 2009-06-01 97.1
```

timeSeries:

```
tX <- tS[-4, ]
tX
```

```
## GMT
##            TS.1 providerCompany* ratingOfOffer*
## 2009-01-01 96.6         UISA Ltd            AAA
## 2009-02-01 96.8         UISA Ltd            AAA
```

```
## 2009-03-01 96.0         UISA Ltd            AAA
## 2009-04-01 95.9      UISA Ltd \n            AAB
## 2009-05-01 96.5      UISA Ltd \n            AAB
## 2009-06-01 97.1      UISA Ltd \n            AAB
```

```
tX@recordIDs
```

```
##   providerCompany ratingOfOffer
## 1        UISA Ltd            AAA
## 2        UISA Ltd            AAA
## 3        UISA Ltd            AAA
## 5     UISA Ltd \n            AAB
## 6     UISA Ltd \n            AAB
## 7     UISA Ltd \n            AAB
```

timeSeries @recordIDs 序列中的所有属性信息，不需要任何附加操作。

2.1.5 时间序列的连接与合并

需加载的 R 包

```
library(zoo)
library(xts)
library(timeSeries)
```

2.1.5.1 如何对两个时间序列对象进行行合并? 使用 rbind 函数，可以完成两个时间序列的行合并。

常规数据

```
data <- c(1:6)
charvec1 <- paste("2009-0", 1:6, "-01", sep = "")
charvec1
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

```
charvec2 <- c(paste("2009-0", 7:9, "-01", sep = ""), paste("2009-",
    10:12, "-01", sep = ""))
charvec2
```

```
## [1] "2009-07-01" "2009-08-01" "2009-09-01"
## [4] "2009-10-01" "2009-11-01" "2009-12-01"
```

zoo:

```
z1 <- zoo(data, as.Date(charvec1))
z2 <- zoo(data + 6, as.Date(charvec2))
rbind(z1, z2)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          4
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
##          5          6          7          8
## 2009-09-01 2009-10-01 2009-11-01 2009-12-01
##          9         10         11         12
```

```
rbind(z2, z1)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          4
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
##          5          6          7          8
## 2009-09-01 2009-10-01 2009-11-01 2009-12-01
##          9         10         11         12
```

注意：rbind 函数的参数输入顺序并不影响函数的操作结果。

xts:

```
x1 <- xts(data, as.Date(charvec1))
x2 <- xts(data + 6, as.Date(charvec2))
rbind(x1, x2)
```

```
## Warning in rbind(deparse.level, ...): mismatched
## types: converting objects to numeric
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
## 2009-07-01    7
## 2009-08-01    8
## 2009-09-01    9
## 2009-10-01   10
## 2009-11-01   11
## 2009-12-01   12
```

```
rbind(x2, x1)
```

```
## Warning in rbind(deparse.level, ...): mismatched
## types: converting objects to numeric
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

```
## 2009-07-01    7
## 2009-08-01    8
## 2009-09-01    9
## 2009-10-01   10
## 2009-11-01   11
## 2009-12-01   12
```

同样地，行合并的结果依然是一个已排序的时间序列。

timeSeries:

```
s1 <- timeSeries(data, charvec1)
s2 <- timeSeries(data + 6, charvec2)
rbind(s1, s2)
```

```
## GMT
##            TS.1_TS.1
## 2009-01-01         1
## 2009-02-01         2
## 2009-03-01         3
## 2009-04-01         4
## 2009-05-01         5
## 2009-06-01         6
## 2009-07-01         7
## 2009-08-01         8
## 2009-09-01         9
## 2009-10-01        10
## 2009-11-01        11
## 2009-12-01        12
```

```
rbind(s2, s1)
```

```
## GMT
##            TS.1_TS.1
```

```
## 2009-07-01         7
## 2009-08-01         8
## 2009-09-01         9
## 2009-10-01        10
## 2009-11-01        11
## 2009-12-01        12
## 2009-01-01         1
## 2009-02-01         2
## 2009-03-01         3
## 2009-04-01         4
## 2009-05-01         5
## 2009-06-01         6
```

注意: rbind 函数的参数输入顺序影响了 rbind 函数的操作结果。不过，这不是 timeSeries 对象的 bug，而是其特色之处。如果想要得到与前面两次相同的结果, 只需要对结果像这样进行排序就可以了。

```
sort(rbind(s2, s1))
```

```
## GMT
##             TS.1_TS.1
## 2009-01-01         1
## 2009-02-01         2
## 2009-03-01         3
## 2009-04-01         4
## 2009-05-01         5
## 2009-06-01         6
## 2009-07-01         7
## 2009-08-01         8
## 2009-09-01         9
## 2009-10-01        10
## 2009-11-01        11
## 2009-12-01        12
```

2.1.5.2 能对时期重叠的时间序列进行行合并吗? 常规数据

```
data1 <- 1:6
data2 <- 3:9
charvec1 <- paste("2009-0", 1:6, "-01", sep = "")
charvec1
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

```
charvec2 <- paste("2009-0", 3:9, "-01", sep = "")
charvec2
```

```
## [1] "2009-03-01" "2009-04-01" "2009-05-01"
## [4] "2009-06-01" "2009-07-01" "2009-08-01"
## [7] "2009-09-01"
```

zoo:

```
z1 <- zoo(data1, as.Date(charvec1))
z2 <- zoo(data2, as.Date(charvec2))
print(try(rbind(z1, z2)))
```

```
## Error in rbind(deparse.level, ...) : indexes overlap
## [1] "Error in rbind(deparse.level, ...) : indexes overlap\n"
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in rbind(deparse.level, ...): indexes overlap>
```

```
print(try(rbind(z2, z1)))
```

```
## Error in rbind(deparse.level, ...) : indexes overlap
## [1] "Error in rbind(deparse.level, ...) : indexes overlap\n"
```

```
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in rbind(deparse.level, ...): indexes overlap>
```

时期重叠的 zoo 格式的时间序列无法进行合并。

xts:

```
x1 <- xts(data1, as.Date(charvec1))
x2 <- xts(data2, as.Date(charvec2))
rbind(x1, x2)
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-03-01    3
## 2009-04-01    4
## 2009-04-01    4
## 2009-05-01    5
## 2009-05-01    5
## 2009-06-01    6
## 2009-06-01    6
## 2009-07-01    7
## 2009-08-01    8
## 2009-09-01    9
```

```
rbind(x2, x1)
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
```

```
## 2009-03-01    3
## 2009-04-01    4
## 2009-04-01    4
## 2009-05-01    5
## 2009-05-01    5
## 2009-06-01    6
## 2009-06-01    6
## 2009-07-01    7
## 2009-08-01    8
## 2009-09-01    9
```

(尽管 rbind 操作顺利地执行了), xts 包不会将时期重叠的时间序列进行简单的合并, 更没有保留原有的时间序列的时间戳顺序，操作结果是一个正序的时间序列。

timeSeries:

```
s1 <- xts(data1, as.Date(charvec1))
s2 <- xts(data2, as.Date(charvec2))
rbind(s1, s2)
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-03-01    3
## 2009-04-01    4
## 2009-04-01    4
## 2009-05-01    5
## 2009-05-01    5
## 2009-06-01    6
## 2009-06-01    6
## 2009-07-01    7
## 2009-08-01    8
## 2009-09-01    9
```

```
rbind(s2, s1)
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-03-01    3
## 2009-04-01    4
## 2009-04-01    4
## 2009-05-01    5
## 2009-05-01    5
## 2009-06-01    6
## 2009-06-01    6
## 2009-07-01    7
## 2009-08-01    8
## 2009-09-01    9
```

timeSeries 对象可以完美支持时期重叠的时间序列的合并操作, 且会保留时间戳的原始顺序。

2.1.5.3 如何对时间序列对象进行列合并? 常规数据

```
data1 <- 1:6
data2 <- data1 + 6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
z1 <- zoo(data1, as.Date(charvec))
z2 <- zoo(data2, as.Date(charvec))
cbind(z1, z2)
```

```
##            z1 z2
## 2009-01-01  1  7
## 2009-02-01  2  8
## 2009-03-01  3  9
## 2009-04-01  4 10
## 2009-05-01  5 11
## 2009-06-01  6 12
```

```
cbind(z2, z1)
```

```
##            z2 z1
## 2009-01-01  7  1
## 2009-02-01  8  2
## 2009-03-01  9  3
## 2009-04-01 10  4
## 2009-05-01 11  5
## 2009-06-01 12  6
```

注意: cbind 函数的参数输入顺序将影响到 cbind 函数的操作结果。

xts:

```
x1 <- xts(data1, as.Date(charvec))
x2 <- xts(data2, as.Date(charvec))
cbind(x1, x2)
```

```
##            x1 x2
## 2009-01-01  1  7
## 2009-02-01  2  8
## 2009-03-01  3  9
```

```
## 2009-04-01  4 10
## 2009-05-01  5 11
## 2009-06-01  6 12
```

```
cbind(x2, x1)
```

```
##            x2 x1
## 2009-01-01  7  1
## 2009-02-01  8  2
## 2009-03-01  9  3
## 2009-04-01 10  4
## 2009-05-01 11  5
## 2009-06-01 12  6
```

timeSeries:

```
s1 <- timeSeries(data1, as.Date(charvec))
s2 <- timeSeries(data2, as.Date(charvec))
cbind(s1, s2)
```

```
## GMT
##            TS.1.1 TS.1.2
## 2009-01-01      1      7
## 2009-02-01      2      8
## 2009-03-01      3      9
## 2009-04-01      4     10
## 2009-05-01      5     11
## 2009-06-01      6     12
```

```
cbind(s2, s1)
```

```
## GMT
##            TS.1.1 TS.1.2
## 2009-01-01      7      1
```

```
## 2009-02-01      8      2
## 2009-03-01      9      3
## 2009-04-01     10      4
## 2009-05-01     11      5
## 2009-06-01     12      6
```

跟 zoo 对象和 xts 对象类似，对 timeSeries 对象进行列合并时，cbind 函数的参数输入顺序会影响到 cbind 函数的操作结果。

2.1.5.4 能对时期重叠的时间序列进行列合并吗? 常规数据

```
data1 <- 1:6
data2 <- 4:8
charvec1 <- paste("2009-0", 1:6, "-01", sep = "")
charvec1
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

```
charvec2 <- paste("2009-0", 4:8, "-01", sep = "")
charvec2
```

```
## [1] "2009-04-01" "2009-05-01" "2009-06-01"
## [4] "2009-07-01" "2009-08-01"
```

zoo:

```
z1 <- zoo(data1, as.Date(charvec1))
z2 <- zoo(data2, as.Date(charvec2))
cbind(z1, z2)
```

```
##            z1 z2
## 2009-01-01  1 NA
## 2009-02-01  2 NA
```

```
## 2009-03-01  3 NA
## 2009-04-01  4  4
## 2009-05-01  5  5
## 2009-06-01  6  6
## 2009-07-01 NA  7
## 2009-08-01 NA  8
```

时期重叠的 zoo 时间序列对象，可以进行列合并。合并过程中产生的缺失值将以 NA 来代替。

xts:

```
x1 <- xts(data1, as.Date(charvec1))
x2 <- xts(data2, as.Date(charvec2))
cbind(x1, x2)
```

```
##            x1 x2
## 2009-01-01  1 NA
## 2009-02-01  2 NA
## 2009-03-01  3 NA
## 2009-04-01  4  4
## 2009-05-01  5  5
## 2009-06-01  6  6
## 2009-07-01 NA  7
## 2009-08-01 NA  8
```

时期重叠的 xts 时间序列对象也可以进行列合并，合并过程中产生的缺失值将以 NA 代替。合并过程中列名会有所变化，注意时间序列的列名丢失了。

timeSeries:

```
s1 <- timeSeries(data1, as.Date(charvec1), units = "s1")
s2 <- timeSeries(data2, as.Date(charvec2), units = "s2")
cbind(s1, s2)
```

```
## GMT
##            s1 s2
## 2009-01-01  1 NA
## 2009-02-01  2 NA
## 2009-03-01  3 NA
## 2009-04-01  4  4
## 2009-05-01  5  5
## 2009-06-01  6  6
## 2009-07-01 NA  7
## 2009-08-01 NA  8
```

timeSeries 对象可以完美支持对时期重叠的时间序列对象进行列合并操作。与 zoo 对象和 xts 对象一样，合并过程中产生的缺失值将以 NA 代替。

2.1.5.5 如何合并两个时间序列对象，合并操作跟列合并和行合并操作有什么不同之处?

base 包中的 merge 函数可以用以合并数据框。

```
args(merge.data.frame)
```

```
## function (x, y, by = intersect(names(x), names(y)), by.x = by,
##     by.y = by, all = FALSE, all.x = all, all.y = all, sort = TRUE,
##     suffixes = c(".x", ".y"), no.dups = TRUE, incomparables = NULL,
##     ...)
## NULL
```

帮助文档里面可以看到如下信息: Merge two data frames by common columns or row names, or do other versions of database "join"operations. 因此，可以直觉推测，merge 函数应该可以像合并数据框一样合并时间序列对象。

注意: 上述推断也意味着针对时间序列对象的合并操作与数据框的合并操作是一样的。

2.1.5.6 将两个一样的时间序列对象合并，会怎样?

常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
z <- zoo(data, as.Date(charvec))
merge(z, z)
```

```
##            z z
## 2009-01-01 1 1
## 2009-02-01 2 2
## 2009-03-01 3 3
## 2009-04-01 4 4
## 2009-05-01 5 5
## 2009-06-01 6 6
```

zoo 对象返回一个双变量时间序列对象。

xts:

```
x <- xts(data, as.Date(charvec))
merge(x, x)
```

```
##            x x.1
## 2009-01-01 1   1
## 2009-02-01 2   2
## 2009-03-01 3   3
## 2009-04-01 4   4
## 2009-05-01 5   5
## 2009-06-01 6   6
```

xts 对象也将返回一个双变量时间序列对象。

timeSeries:

```
s <- timeSeries(data, charvec)
merge(s, s)
```

```
## GMT
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

```
merge(as.data.frame(s), as.data.frame(s))
```

```
##   TS.1
## 1    1
## 2    2
## 3    3
## 4    4
## 5    5
## 6    6
```

timeSeries 对象的操作方式与 zoo 对象和 xts 对象不同，timeSeries 对象跟 data.frame 对象更类似。

2.1.5.7 如何合并两个不同的单变量时间序列? 常规数据

```
data1 <- 1:6
data2 <- data1 + 3
charvec1 <- paste("2009-0", 1:6, "-01", sep = "")
charvec1
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

```
charvec2 <- paste("2009-0", 4:9, "-01", sep = "")
charvec2
```

```
## [1] "2009-04-01" "2009-05-01" "2009-06-01"
## [4] "2009-07-01" "2009-08-01" "2009-09-01"
```

zoo:

```
z1 <- zoo(data1, as.Date(charvec1))
z2 <- zoo(data2, as.Date(charvec2))
merge(z1, z2)
```

```
##            z1 z2
## 2009-01-01  1 NA
## 2009-02-01  2 NA
## 2009-03-01  3 NA
## 2009-04-01  4  4
## 2009-05-01  5  5
## 2009-06-01  6  6
## 2009-07-01 NA  7
## 2009-08-01 NA  8
## 2009-09-01 NA  9
```

xts:

```
x1 <- xts(data1, as.Date(charvec1))
x2 <- xts(data2, as.Date(charvec2))
merge(x1, x2)
```

```
##            x1 x2
## 2009-01-01  1 NA
```

```
## 2009-02-01  2 NA
## 2009-03-01  3 NA
## 2009-04-01  4  4
## 2009-05-01  5  5
## 2009-06-01  6  6
## 2009-07-01 NA  7
## 2009-08-01 NA  8
## 2009-09-01 NA  9
```

timeSeries:

```
s1 <- timeSeries(data1, as.Date(charvec1), units = "s1")
s2 <- timeSeries(data2, as.Date(charvec2), units = "s2")
merge(s1, s2)
```

```
## GMT
##            s1 s2
## 2009-01-01  1 NA
## 2009-02-01  2 NA
## 2009-03-01  3 NA
## 2009-04-01  4  4
## 2009-05-01  5  5
## 2009-06-01  6  6
## 2009-07-01 NA  7
## 2009-08-01 NA  8
## 2009-09-01 NA  9
```

三种类型的时间序列对象的合并结果都一样。

2.1.5.8 把两个含有相同信息的单变量时间序列合并在一起，会如何？ 常规数据

```
data1 <- 1:6
data2 <- data1 + 3
charvec1 <- paste("2009-0", 1:6, "-01", sep = "")
charvec1
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

```
charvec2 <- paste("2009-0", 4:9, "-01", sep = "")
charvec2
```

```
## [1] "2009-04-01" "2009-05-01" "2009-06-01"
## [4] "2009-07-01" "2009-08-01" "2009-09-01"
```

zoo:

无法完成该操作。

xts:

对于 xts 时间序列数据类型, 可以现将列名称设置成一致的, 然后将其合并在一起。

```
x1 <- xts(data1, as.Date(charvec1))
colnames(x1) <- "x"
z2 <- xts(data2, as.Date(charvec2))
colnames(x2) <- "x"
merge(x1, x2)
```

```
##             x x.1
## 2009-01-01  1  NA
## 2009-02-01  2  NA
## 2009-03-01  3  NA
## 2009-04-01  4   4
## 2009-05-01  5   5
```

```
## 2009-06-01  6   6
## 2009-07-01 NA   7
## 2009-08-01 NA   8
## 2009-09-01 NA   9
```

操作返回一个列名称为 x 和 x.1 的双变量时间序列。

timeSeries:

```
s1 <- timeSeries(data1, charvec1, units = "s")
s2 <- timeSeries(data2, charvec2, units = "s")
merge(s1, s2)
```

```
## GMT
##            s
## 2009-01-01 1
## 2009-02-01 2
## 2009-03-01 3
## 2009-04-01 4
## 2009-05-01 5
## 2009-06-01 6
## 2009-07-01 7
## 2009-08-01 8
## 2009-09-01 9
```

在 timeSeries 中, 我们得到了不一样的结果。因为两个序列含有相同的信息集合 “s”, 我们所得到的是一个单变量序列。

2.1.5.9 能将时间序列对象跟一个数值元素合并吗? 常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

```
const <- 3.4
```

zoo:

```
z <- zoo(data, as.Date(charvec))
merge(z, const)
```

```
##            z const
## 2009-01-01 1   3.4
## 2009-02-01 2   3.4
## 2009-03-01 3   3.4
## 2009-04-01 4   3.4
## 2009-05-01 5   3.4
## 2009-06-01 6   3.4
```

xts:

```
x <- xts(data, as.Date(charvec))
merge(x, const)
```

```
##            x const
## 2009-01-01 1   3.4
## 2009-02-01 2   3.4
## 2009-03-01 3   3.4
## 2009-04-01 4   3.4
## 2009-05-01 5   3.4
## 2009-06-01 6   3.4
```

timeSeries:

```
s <- timeSeries(data, charvec)
merge(s, const)
```

```
## GMT
```

```
##            TS.1   s
## 2009-01-01    1 3.4
## 2009-02-01    2 3.4
## 2009-03-01    3 3.4
## 2009-04-01    4 3.4
## 2009-05-01    5 3.4
## 2009-06-01    6 3.4
```

2.1.5.10 能将时间序列对象和数值型向量合并吗? 常规数据

```
data <- 1:6
data
```

```
## [1] 1 2 3 4 5 6
```

```
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

```
vec <- 3.4 - 1:6
vec
```

```
## [1]  2.4  1.4  0.4 -0.6 -1.6 -2.6
```

zoo:

```
z <- zoo(data, as.Date(charvec))
merge(z, vec)
```

```
##            z  vec
## 2009-01-01 1  2.4
## 2009-02-01 2  1.4
```

```
## 2009-03-01 3  0.4
## 2009-04-01 4 -0.6
## 2009-05-01 5 -1.6
## 2009-06-01 6 -2.6
```

xts:

```
x <- xts(data, as.Date(charvec))
merge(x, vec)
```

```
##            x  vec
## 2009-01-01 1  2.4
## 2009-02-01 2  1.4
## 2009-03-01 3  0.4
## 2009-04-01 4 -0.6
## 2009-05-01 5 -1.6
## 2009-06-01 6 -2.6
```

timeSeries:

```
s <- timeSeries(data, charvec)
merge(s, vec)
```

```
## GMT
##            TS.1    s
## 2009-01-01    1  2.4
## 2009-02-01    2  1.4
## 2009-03-01    3  0.4
## 2009-04-01    4 -0.6
## 2009-05-01    5 -1.6
## 2009-06-01    6 -2.6
```

2.1.5.11 能将时间序列对象与数值型矩阵合并吗? 常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

```
mat <- matrix((1:12) - 6, ncol = 2) - 3.4
mat
```

```
##      [,1] [,2]
## [1,] -8.4 -2.4
## [2,] -7.4 -1.4
## [3,] -6.4 -0.4
## [4,] -5.4  0.6
## [5,] -4.4  1.6
## [6,] -3.4  2.6
```

zoo:

```
z <- zoo(data, as.Date(charvec))
merge(z, mat)
```

```
##            z mat.1 mat.2
## 2009-01-01 1  -8.4  -2.4
## 2009-02-01 2  -7.4  -1.4
## 2009-03-01 3  -6.4  -0.4
## 2009-04-01 4  -5.4   0.6
## 2009-05-01 5  -4.4   1.6
## 2009-06-01 6  -3.4   2.6
```

xts:

```
x <- xts(data, as.Date(charvec))
merge(x, mat)
```

```
##            x  mat mat.1
## 2009-01-01 1 -8.4  -2.4
## 2009-02-01 2 -7.4  -1.4
## 2009-03-01 3 -6.4  -0.4
## 2009-04-01 4 -5.4   0.6
## 2009-05-01 5 -4.4   1.6
## 2009-06-01 6 -3.4   2.6
```

timeSeries:

```
s <- timeSeries(data, charvec)
merge(s, mat)
```

```
## GMT
##            TS.1 mat.1 mat.2
## 2009-01-01    1  -8.4  -2.4
## 2009-02-01    2  -7.4  -1.4
## 2009-03-01    3  -6.4  -0.4
## 2009-04-01    4  -5.4   0.6
## 2009-05-01    5  -4.4   1.6
## 2009-06-01    6  -3.4   2.6
```

2.1.6 时间序列对象的取子集操作

需要加载的 R 包

```
library(zoo)
library(xts)
library(timeSeries)
```

2.1.6.1 如何对向量和矩阵进行取子集操作? 向量是线性对象，因此只需要一个下标即可对其进行取子集操作。

```
vec <- rnorm(6)
vec
```

```
## [1] -0.41538239  0.81159149  0.26277553
## [4] -0.02925881  1.04997825  1.31966758
```

```
vec[3:4]
```

```
## [1]  0.26277553 -0.02925881
```

向量对象没有维度，其长度等于其所含有的元素数。

```
dim(vec)
```

```
## NULL
```

```
length(vec)
```

```
## [1] 6
```

矩阵是一个方形的对象, 故需要一对下标对其进行取子集操作：一个为行标, 一个为列标。

```
mat <- matrix(rnorm(18), ncol = 3)
mat
```

```
##             [,1]        [,2]       [,3]
## [1,]  0.8348494  0.07607636  0.5529988
## [2,]  0.7920079  1.41468159 -1.1289432
## [3,] -0.4591048 -0.76013552 -0.6328236
## [4,]  0.8524197 -0.68364284  0.8494884
## [5,]  0.2848695  0.16952424  1.5089333
## [6,] -0.5056958  0.69517590  1.2011990
```

```
mat[3:4, ]
```

```
##            [,1]       [,2]       [,3]
## [1,] -0.4591048 -0.7601355 -0.6328236
## [2,]  0.8524197 -0.6836428  0.8494884
```

```
mat[, 2:3]
```

```
##             [,1]       [,2]
## [1,]  0.07607636  0.5529988
## [2,]  1.41468159 -1.1289432
## [3,] -0.76013552 -0.6328236
## [4,] -0.68364284  0.8494884
## [5,]  0.16952424  1.5089333
## [6,]  0.69517590  1.2011990
```

```
mat[3:4, 2:3]
```

```
##            [,1]       [,2]
## [1,] -0.7601355 -0.6328236
## [2,] -0.6836428  0.8494884
```

对于矩阵而言，有如下结果：

```
dim(mat)
```

```
## [1] 6 3
```

```
length(mat)
```

```
## [1] 18
```

dim 函数返回了矩阵的行列数。length 函数返回了矩阵中元素的总数。当我们只取矩阵中的一行或一列会怎么样?

```
mat[3, ]
```

```
## [1] -0.4591048 -0.7601355 -0.6328236
```

```
mat[, 2]
```

```
## [1]  0.07607636  1.41468159 -0.76013552
## [4] -0.68364284  0.16952424  0.69517590
```

这时长方形的对象变为线性的, 结果变成了一个 (单维) 向量 (而不是二维矩阵了)。为了避免这一情况的发生, 我们可以采取一些措施, 以确保我们不丢掉维度信息。

```
rowmat <- mat[3, , drop = FALSE]
colmat <- mat[, 2, drop = FALSE]
```

2.1.6.2 对时间序列对象进行取子集操作时，跟对向量和矩阵操作一样吗?

不一定。取决于时间序列的对象类型。

常规数据

```
data1 <- rnorm(1:6)
data2 <- matrix(rnorm(18), ncol = 3)
colnames(data2) <- LETTERS[1:3]
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

单变量时间序列

```
z <- zoo(data1, as.Date(charvec))
z[3:4]
```

```
## 2009-03-01 2009-04-01
## -1.4461873 -0.9916621
```

多变量时间序列

```
Z <- zoo(data2, as.Date(charvec))
Z[3:4, ]
```

```
##                      A         B          C
## 2009-03-01 -1.18503011 1.6696444 -0.9093841
## 2009-04-01  0.04302976 0.6042475  0.5437734
```

```
Z[, 2:3]
```

```
##                     B          C
## 2009-01-01  0.3592784 -0.7627196
## 2009-02-01 -1.9737346 -0.6539729
## 2009-03-01  1.6696444 -0.9093841
## 2009-04-01  0.6042475  0.5437734
## 2009-05-01  0.7925452 -0.0368513
## 2009-06-01 -0.8160521 -0.3216701
```

```
Z[3:4, 2:3]
```

```
##                    B          C
## 2009-03-01 1.6696444 -0.9093841
## 2009-04-01 0.6042475  0.5437734
```

```
Z[, 2:3]
```

```
##                     B          C
## 2009-01-01  0.3592784 -0.7627196
## 2009-02-01 -1.9737346 -0.6539729
## 2009-03-01  1.6696444 -0.9093841
## 2009-04-01  0.6042475  0.5437734
## 2009-05-01  0.7925452 -0.0368513
## 2009-06-01 -0.8160521 -0.3216701
```

```
Z[3:4, 2:3]
```

```
##                     B          C
## 2009-03-01 1.6696444 -0.9093841
## 2009-04-01 0.6042475  0.5437734
```

要注意的是, 一个单变量的 zoo 时间序列对象就像一个向量, 而一个多变量的 zoo 时间序列对象则如同一个矩阵一样。取单独的行或列都会丢失维度(参见前一节)。

xts:

单变量时间序列

```
x <- xts(data1, as.Date(charvec))
x[3:4]
```

```
##                   [,1]
## 2009-03-01 -1.4461873
## 2009-04-01 -0.9916621
```

多变量时间序列

```
X <- xts(data2, as.Date(charvec))
X[3:4, ]
```

```
##                          A         B          C
## 2009-03-01 -1.18503011 1.6696444 -0.9093841
## 2009-04-01  0.04302976 0.6042475  0.5437734
```

```
X[, 2:3]
```

```
##                          B          C
## 2009-01-01  0.3592784 -0.7627196
## 2009-02-01 -1.9737346 -0.6539729
## 2009-03-01  1.6696444 -0.9093841
## 2009-04-01  0.6042475  0.5437734
## 2009-05-01  0.7925452 -0.0368513
## 2009-06-01 -0.8160521 -0.3216701
```

```
X[3:4, 2:3]
```

```
##                        B          C
## 2009-03-01 1.6696444 -0.9093841
## 2009-04-01 0.6042475  0.5437734
```

```
X[, 2:3]
```

```
##                          B          C
## 2009-01-01  0.3592784 -0.7627196
## 2009-02-01 -1.9737346 -0.6539729
## 2009-03-01  1.6696444 -0.9093841
## 2009-04-01  0.6042475  0.5437734
## 2009-05-01  0.7925452 -0.0368513
## 2009-06-01 -0.8160521 -0.3216701
```

```
X[3:4, 2:3]
```

```
##                    B          C
## 2009-03-01 1.6696444 -0.9093841
## 2009-04-01 0.6042475  0.5437734
```

xts 对象通常被默认为方形对象来处理, 即便在单变量的情况下也是如此。这种独特的处理方式是大大方便了对金融时间序列取子集操作。

timeSeries:

单变量时间序列

```
s <- timeSeries(data1, charvec)
s[3:4]
```

```
## [1] -1.4461873 -0.9916621
```

```
s[3:4, ]
```

```
## GMT
##                   TS.1
## 2009-03-01 -1.4461873
## 2009-04-01 -0.9916621
```

多变量时间序列

```
S <- timeSeries(data2, charvec)
S[3:4, ]
```

```
## GMT
##                         A         B          C
## 2009-03-01 -1.18503011 1.6696444 -0.9093841
## 2009-04-01  0.04302976 0.6042475  0.5437734
```

```
S[, 2:3]
```

```
## GMT
##                      B          C
## 2009-01-01  0.3592784 -0.7627196
## 2009-02-01 -1.9737346 -0.6539729
## 2009-03-01  1.6696444 -0.9093841
## 2009-04-01  0.6042475  0.5437734
## 2009-05-01  0.7925452 -0.0368513
## 2009-06-01 -0.8160521 -0.3216701
```

```
S[3:4, 2:3]
```

```
## GMT
##                     B          C
## 2009-03-01 1.6696444 -0.9093841
## 2009-04-01 0.6042475  0.5437734
```

注意：timeSeries 对象总是需要用一对下标来完成取子集操作。这是 timeSeries 的一个特色。

```
S[, 2:3]
```

```
## GMT
##                      B          C
## 2009-01-01  0.3592784 -0.7627196
## 2009-02-01 -1.9737346 -0.6539729
## 2009-03-01  1.6696444 -0.9093841
## 2009-04-01  0.6042475  0.5437734
## 2009-05-01  0.7925452 -0.0368513
## 2009-06-01 -0.8160521 -0.3216701
```

```
S[3:4, 2:3]
```

```
## GMT
##                      B          C
## 2009-03-01 1.6696444 -0.9093841
## 2009-04-01 0.6042475  0.5437734
```

timeSeries 对象跟 xts 对象类似，都是方形对象。操作上又不少共同之处。

2.1.6.3 能否用列名对多变量时间序列进行取子集操作? 常规数据

```
data <- matrix(rnorm(18), ncol = 3)
colnames(data) <- LETTERS[1:3]
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
Z[, "A"]
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
## -0.4186377 -0.8865769 -0.7116370  1.1170256
## 2009-05-01 2009-06-01
##  0.7232390  0.4596303
```

xts:

```
X <- xts(data, as.Date(charvec))
X[, "A"]
```

```
##                     A
## 2009-01-01 -0.4186377
## 2009-02-01 -0.8865769
## 2009-03-01 -0.7116370
## 2009-04-01  1.1170256
## 2009-05-01  0.7232390
## 2009-06-01  0.4596303
```

timeSeries:

```
S <- timeSeries(data, charvec)
S[, "A"]
```

```
## GMT
##                     A
## 2009-01-01 -0.4186377
## 2009-02-01 -0.8865769
## 2009-03-01 -0.7116370
## 2009-04-01  1.1170256
## 2009-05-01  0.7232390
## 2009-06-01  0.4596303
```

2.1.6.4 能否用 $ 符对多变量时间序列进行取列子集的操作? 常规数据

```
data <- matrix(rnorm(18), ncol = 3)
colnames(data) <- LETTERS[1:3]
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
Z$B
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
## -1.1259095 -0.8096899  0.6392114 -1.4089112
## 2009-05-01 2009-06-01
## -1.0756599 -0.7133154
```

xts:

```
X <- xts(data, as.Date(charvec))
X$B
```

```
##                    B
## 2009-01-01 -1.1259095
## 2009-02-01 -0.8096899
## 2009-03-01  0.6392114
## 2009-04-01 -1.4089112
## 2009-05-01 -1.0756599
## 2009-06-01 -0.7133154
```

timeSeries:

```
S <- timeSeries(data, charvec)
S$B
```

```
## [1] -1.1259095 -0.8096899  0.6392114 -1.4089112
## [5] -1.0756599 -0.7133154
```

2.1.6.5 能否通过字符串时间戳对多变量时间序列对象进行取子集操作?

常规数据

```
data <- matrix(rnorm(18), ncol = 3)
colnames(data) <- LETTERS[1:3]
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
charvec[3:4]
```

```
## [1] "2009-03-01" "2009-04-01"
```

```
Z[charvec[3:4], ]
```

```
##                     A          B          C
## 2009-03-01 -0.6071183 -1.0086160  1.1936528
## 2009-04-01 -2.0975569  0.9087197 -0.3568767
```

xts:

```
X <- xts(data, as.Date(charvec))
charvec[3:4]
```

```
## [1] "2009-03-01" "2009-04-01"
```

```
X[charvec[3:4], ]
```

```
##                     A          B          C
## 2009-03-01 -0.6071183 -1.0086160  1.1936528
## 2009-04-01 -2.0975569  0.9087197 -0.3568767
```

timeSeries:

```
S <- timeSeries(data, charvec)
charvec[3:4]
```

```
## [1] "2009-03-01" "2009-04-01"
```

```
S[charvec[3:4], ]
```

```
## GMT
##                      A          B          C
## 2009-03-01 -0.6071183 -1.0086160  1.1936528
## 2009-04-01 -2.0975569  0.9087197 -0.3568767
```

2.1.6.6 能否用多变量时间序列自身的时间戳对其进行取子集操作呢?

常规数据

```
data <- matrix(rnorm(18), ncol = 3)
colnames(data) <- LETTERS[1:3]
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
timeStamp <- index(Z)[3:4]
timeStamp
```

```
## [1] "2009-03-01" "2009-04-01"
```

```
Z[timeStamp, ]
```

```
##                     A          B         C
## 2009-03-01 1.1525930 -0.2007126  0.535684
## 2009-04-01 0.2062312 -0.3528623 -1.247634
```

xts:

```
X <- xts(data, as.Date(charvec))
timeStamp <- index(X)[3:4]
timeStamp
```

```
## [1] "2009-03-01" "2009-04-01"
```

```
X[timeStamp, ]
```

```
##                     A          B         C
## 2009-03-01 1.1525930 -0.2007126  0.535684
## 2009-04-01 0.2062312 -0.3528623 -1.247634
```

timeSeries:

```
S <- timeSeries(data, charvec)
timeStamp <- time(S)[3:4]
timeStamp
```

```
## GMT
## [1] [2009-03-01] [2009-04-01]
```

```
S[timeStamp, ]
```

```
## GMT
##                     A          B         C
## 2009-03-01 1.1525930 -0.2007126  0.535684
## 2009-04-01 0.2062312 -0.3528623 -1.247634
```

2.1.6.7 能否用逻辑操作对多变量时间序列对象进行取子集操作? 常规数据

```
data <- matrix(rnorm(18), ncol = 3)
colnames(data) <- LETTERS[1:3]
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
timeStamp <- index(Z) > index(Z)[3]
timeStamp
```

```
## [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

```
Z[timeStamp, ]
```

```
##                    A          B          C
## 2009-04-01 -0.4879470 -0.6855596  0.8901843
## 2009-05-01 -0.5992318  1.7415832 -0.3190306
## 2009-06-01 -1.1026455 -0.5636855 -1.3224028
```

xts:

```
X <- xts(data, as.Date(charvec))
timeStamp <- index(X) > index(X)[3]
timeStamp
```

```
## [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

```
X[timeStamp, ]
```

```
##                      A          B          C
## 2009-04-01 -0.4879470 -0.6855596  0.8901843
## 2009-05-01 -0.5992318  1.7415832 -0.3190306
## 2009-06-01 -1.1026455 -0.5636855 -1.3224028
```

timeSeries:

```
S <- timeSeries(data, charvec)
timeStamp <- time(S) > time(S)[3]
timeStamp
```

```
## [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

```
S[timeStamp, ]
```

```
## GMT
##                      A          B          C
## 2009-04-01 -0.4879470 -0.6855596  0.8901843
## 2009-05-01 -0.5992318  1.7415832 -0.3190306
## 2009-06-01 -1.1026455 -0.5636855 -1.3224028
```

2.1.6.8 如何提取序列的起始和终止日期? 常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
z <- zoo(data, as.Date(charvec))
c(start(x), end(x))
```

```
## [1] "2009-01-01" "2009-06-01"
```

xts:

```
x <- xts(data, as.Date(charvec))
c(start(x), end(x))
```

```
## [1] "2009-01-01" "2009-06-01"
```

timeSeries:

```
s <- timeSeries(data, charvec)
c(start(s), end(s))
```

```
## GMT
## [1] [2009-01-01] [2009-06-01]
```

由于 timeSeries 对象支持不排序的时间戳，因此序列中的首末记录跟正序的时间戳记录未必相同，见下例：

```
s <- timeSeries(data, sample(charvec))
c(start(s), end(s))
```

```
## GMT
## [1] [2009-01-01] [2009-06-01]
```

```
c(time(s[1, ]), time(s[6, ]))
```

```
## GMT
## [1] [2009-04-01] [2009-06-01]
```

2.1.7 时间序列对象与类函数

需要加载的 R 包。

```
library(zoo)
library(xts)
library(timeSeries)
```

首先，看一下 base 包和 methods 包中关于 groupGeneric 函数的帮助文档。

S3 类基础函数，可以大致分为四类：数学、Ops 类、描述类和复数类。

- Math(x, …)
- Ops(e1, e2)
- Complex(z)
- Summary(…, na.rm = FALSE)

S4 类基础函数可以大概分为八类：

- Arith(e1, e2)
- Compare(e1, e2)
- Ops(e1, e2)
- Logic(e1, e2)
- Math(x)
- Math2(x, digits)
- Summary(x, …, na.rm = FALSE)
- Complex(z)

从中选择几个与金融时间序列分析相关的函数进行演示。演示的数据对象是多变量时间序列和单变量时间序列。

常规数据

```
data <- matrix(runif(18), ncol = 3)
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
colnames(Z) <- paste("Z", 1:3, sep = ".")
Z
```

```
##                   Z.1         Z.2        Z.3
## 2009-01-01 0.78197354 0.917210460 0.83615406
## 2009-02-01 0.06460017 0.004483941 0.24073148
## 2009-03-01 0.78063172 0.866422171 0.06695467
## 2009-04-01 0.25625164 0.894887728 0.04684698
## 2009-05-01 0.58594426 0.844396831 0.74879991
## 2009-06-01 0.59209478 0.935597168 0.21707589
```

xts:

```
X <- xts(data, as.Date(charvec))
colnames(X) <- paste("X", 1:3, sep = ".")
X
```

```
##                   X.1         X.2        X.3
## 2009-01-01 0.78197354 0.917210460 0.83615406
## 2009-02-01 0.06460017 0.004483941 0.24073148
## 2009-03-01 0.78063172 0.866422171 0.06695467
## 2009-04-01 0.25625164 0.894887728 0.04684698
## 2009-05-01 0.58594426 0.844396831 0.74879991
## 2009-06-01 0.59209478 0.935597168 0.21707589
```

timeSeries:

```
S <- timeSeries(data, charvec)
colnames(S) <- paste("S", 1:3, sep = ".")
S
```

```
## GMT
##                   S.1         S.2        S.3
## 2009-01-01 0.78197354 0.917210460 0.83615406
## 2009-02-01 0.06460017 0.004483941 0.24073148
## 2009-03-01 0.78063172 0.866422171 0.06695467
## 2009-04-01 0.25625164 0.894887728 0.04684698
## 2009-05-01 0.58594426 0.844396831 0.74879991
## 2009-06-01 0.59209478 0.935597168 0.21707589
```

zoo:

```
z <- Z[, 1]
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
## 0.78197354 0.06460017 0.78063172 0.25625164
## 2009-05-01 2009-06-01
## 0.58594426 0.59209478
```

xts:

```
x <- X[, 1]
x
```

```
##                   X.1
## 2009-01-01 0.78197354
## 2009-02-01 0.06460017
## 2009-03-01 0.78063172
## 2009-04-01 0.25625164
## 2009-05-01 0.58594426
## 2009-06-01 0.59209478
```

timeSeries:

```
s <- S[, 1]
s
```

```
## GMT
##                    S.1
## 2009-01-01 0.78197354
## 2009-02-01 0.06460017
## 2009-03-01 0.78063172
## 2009-04-01 0.25625164
## 2009-05-01 0.58594426
## 2009-06-01 0.59209478
```

2.1.7.1 时间序列对象支持哪些四则运算符？ “+”, “-”, “*“,”“,”%%“,”%/%“,”/”

.

zoo:

```
Z[, 1] + Z[, 2]
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
## 1.69918400 0.06908411 1.64705389 1.15113937
## 2009-05-01 2009-06-01
## 1.43034109 1.52769195
```

xts:

```
X[, 1] + X[, 2]
```

```
##                    X.1
## 2009-01-01 1.69918400
## 2009-02-01 0.06908411
## 2009-03-01 1.64705389
```

```
## 2009-04-01 1.15113937
## 2009-05-01 1.43034109
## 2009-06-01 1.52769195
```

timeSeries:

```
S[, 1] + S[, 2]
```

```
## GMT
##                    S.1
## 2009-01-01 1.69918400
## 2009-02-01 0.06908411
## 2009-03-01 1.64705389
## 2009-04-01 1.15113937
## 2009-05-01 1.43034109
## 2009-06-01 1.52769195
```

2.1.7.2 时间序列对象支持哪些比较运算符?

“==”, “>”, “<”, “!=”, “<=”, “>=”

zoo:

```
Z[, 1] > Z[, 2]
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##      FALSE       TRUE      FALSE      FALSE
## 2009-05-01 2009-06-01
##      FALSE      FALSE
```

xts:

```
X[, 1] > X[, 2]
```

```
##              X.1
```

```
## 2009-01-01 FALSE
## 2009-02-01  TRUE
## 2009-03-01 FALSE
## 2009-04-01 FALSE
## 2009-05-01 FALSE
## 2009-06-01 FALSE
```

timeSeries:

```
S[, 1] > S[, 2]
```

```
## GMT
##               S.1
## 2009-01-01 FALSE
## 2009-02-01  TRUE
## 2009-03-01 FALSE
## 2009-04-01 FALSE
## 2009-05-01 FALSE
## 2009-06-01 FALSE
```

2.1.7.3 时间序列对象支持哪些逻辑运算? “&”, “|”

zoo:

```
(Z[, 1] > Z[, 2]) & (Z[, 2] < Z[, 3])
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##      FALSE       TRUE      FALSE      FALSE
## 2009-05-01 2009-06-01
##      FALSE      FALSE
```

xts:

```
(X[, 1] > X[, 2]) & (X[, 2] < X[, 3])
```

```
##              X.1
## 2009-01-01 FALSE
## 2009-02-01  TRUE
## 2009-03-01 FALSE
## 2009-04-01 FALSE
## 2009-05-01 FALSE
## 2009-06-01 FALSE
```

timeSeries:

```
(S[, 1] > S[, 2]) & (S[, 2] < S[, 3])
```

```
## GMT
##              S.1
## 2009-01-01 FALSE
## 2009-02-01  TRUE
## 2009-03-01 FALSE
## 2009-04-01 FALSE
## 2009-05-01 FALSE
## 2009-06-01 FALSE
```

2.1.7.4 时间序列对象支持哪些系统操作？ “Arith”, “Compare”, “Logic”

2.1.7.5 时间序列对象支持哪些数学运算？ “abs”, “sign”, “sqrt”, “ceiling”, “floor”, “trunc”, “cummax”, “cummin”, “cumprod”, “cumsum”, “log”, “log10”, “log2”, “log1p”, “acos”, “acosh”, “asin”, “asinh”, “atan”, “atanh”, “exp”, “expm1”, “cos”, “cosh”, “sin”, “sinh”, “tan”, “tanh”, “gamma”, “lgamma”, “digamma”, “trigamma”

zoo:

```
log(abs(Z))
```

```
##                   Z.1         Z.2        Z.3
## 2009-01-01 -0.2459344 -0.08641832 -0.1789424
## 2009-02-01 -2.7395382 -5.40725303 -1.4240732
## 2009-03-01 -0.2476518 -0.14338299 -2.7037395
## 2009-04-01 -1.3615953 -0.11105701 -3.0608688
## 2009-05-01 -0.5345306 -0.16913272 -0.2892835
## 2009-06-01 -0.5240886 -0.06657027 -1.5275083
```

xts:

```
log(abs(X))
```

```
##                   X.1         X.2        X.3
## 2009-01-01 -0.2459344 -0.08641832 -0.1789424
## 2009-02-01 -2.7395382 -5.40725303 -1.4240732
## 2009-03-01 -0.2476518 -0.14338299 -2.7037395
## 2009-04-01 -1.3615953 -0.11105701 -3.0608688
## 2009-05-01 -0.5345306 -0.16913272 -0.2892835
## 2009-06-01 -0.5240886 -0.06657027 -1.5275083
```

timeSeries:

```
log(abs(S))
```

```
## GMT
##                   S.1         S.2        S.3
## 2009-01-01 -0.2459344 -0.08641832 -0.1789424
## 2009-02-01 -2.7395382 -5.40725303 -1.4240732
## 2009-03-01 -0.2476518 -0.14338299 -2.7037395
## 2009-04-01 -1.3615953 -0.11105701 -3.0608688
## 2009-05-01 -0.5345306 -0.16913272 -0.2892835
## 2009-06-01 -0.5240886 -0.06657027 -1.5275083
```

2.1.7.6 时间序列对象支持哪些 Math2 类运算? "round", "signif"

zoo:

```
round(Z, 2)
```

```
##             Z.1  Z.2  Z.3
## 2009-01-01 0.78 0.92 0.84
## 2009-02-01 0.06 0.00 0.24
## 2009-03-01 0.78 0.87 0.07
## 2009-04-01 0.26 0.89 0.05
## 2009-05-01 0.59 0.84 0.75
## 2009-06-01 0.59 0.94 0.22
```

```
signif(Z, 2)
```

```
##              Z.1    Z.2   Z.3
## 2009-01-01 0.780 0.9200 0.840
## 2009-02-01 0.065 0.0045 0.240
## 2009-03-01 0.780 0.8700 0.067
## 2009-04-01 0.260 0.8900 0.047
## 2009-05-01 0.590 0.8400 0.750
## 2009-06-01 0.590 0.9400 0.220
```

xts:

```
round(X, 2)
```

```
##             X.1  X.2  X.3
## 2009-01-01 0.78 0.92 0.84
## 2009-02-01 0.06 0.00 0.24
## 2009-03-01 0.78 0.87 0.07
## 2009-04-01 0.26 0.89 0.05
## 2009-05-01 0.59 0.84 0.75
## 2009-06-01 0.59 0.94 0.22
```

```
signif(X, 2)
```

```
##              X.1    X.2   X.3
## 2009-01-01 0.780 0.9200 0.840
## 2009-02-01 0.065 0.0045 0.240
## 2009-03-01 0.780 0.8700 0.067
## 2009-04-01 0.260 0.8900 0.047
## 2009-05-01 0.590 0.8400 0.750
## 2009-06-01 0.590 0.9400 0.220
```

timeSeries:

```
round(S, 2)
```

```
## GMT
##             S.1  S.2  S.3
## 2009-01-01 0.78 0.92 0.84
## 2009-02-01 0.06 0.00 0.24
## 2009-03-01 0.78 0.87 0.07
## 2009-04-01 0.26 0.89 0.05
## 2009-05-01 0.59 0.84 0.75
## 2009-06-01 0.59 0.94 0.22
```

```
signif(S, 2)
```

```
## GMT
##              S.1    S.2   S.3
## 2009-01-01 0.780 0.9200 0.840
## 2009-02-01 0.065 0.0045 0.240
## 2009-03-01 0.780 0.8700 0.067
## 2009-04-01 0.260 0.8900 0.047
## 2009-05-01 0.590 0.8400 0.750
## 2009-06-01 0.590 0.9400 0.220
```

2.1.7.7 时间序列对象支持哪些“统计描述”运算? “max”, “min”, “range”, “prod”, “sum”, “any”, “all”

zoo:

```
max(z)
```

```
## [1] 0.7819735
```

```
min(z)
```

```
## [1] 0.06460017
```

```
range(z)
```

```
## [1] 0.06460017 0.78197354
```

```
prod(z)
```

```
## [1] 0.003505792
```

```
sum(z)
```

```
## [1] 3.061496
```

xts:

```
max(x)
```

```
## [1] 0.7819735
```

```
min(x)
```

```
## [1] 0.06460017
```

```
range(x)
```

```
## [1] 0.06460017 0.78197354
```

```
prod(x)
```

```
## [1] 0.003505792
```

```
sum(x)
```

```
## [1] 3.061496
```

timeSeries:

```
max(s)
```

```
## [1] 0.7819735
```

```
min(s)
```

```
## [1] 0.06460017
```

```
range(s)
```

```
## [1] 0.06460017 0.78197354
```

```
prod(s)
```

```
## [1] 0.003505792
```

```
sum(s)
```

```
## [1] 3.061496
```

2.1.7.8 时间序列对象支持哪些复数运算? “Arg”, “Conj”, “Im”, “Mod”, “Re”

```
u <- c(0, 0+2i)
u
```

```
## [1] 0+0i 0+2i
```

zoo:

```
Arg(z + u)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##   0.000000   1.538507   0.000000   1.443365
## 2009-05-01 2009-06-01
##   0.000000   1.282970
```

```
Conj(z + u)
```

```
##    2009-01-01    2009-02-01    2009-03-01
## 0.78197354+0i 0.06460017-2i 0.78063172+0i
##    2009-04-01    2009-05-01    2009-06-01
## 0.25625164-2i 0.58594426+0i 0.59209478-2i
```

```
Im(z + u)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          0          2          0          2
## 2009-05-01 2009-06-01
##          0          2
```

```
Mod(z + u)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.7819735  2.0010430  0.7806317  2.0163494
## 2009-05-01 2009-06-01
##  0.5859443  2.0858035
```

```
Re(z + u)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
## 0.78197354 0.06460017 0.78063172 0.25625164
## 2009-05-01 2009-06-01
## 0.58594426 0.59209478
```

xts:

```
Arg(x + u)
```

```
##                 X.1
## 2009-01-01 0.000000
## 2009-02-01 1.538507
## 2009-03-01 0.000000
## 2009-04-01 1.443365
## 2009-05-01 0.000000
## 2009-06-01 1.282970
```

```
Conj(x + u)
```

```
##                      X.1
## 2009-01-01 0.78197354+0i
## 2009-02-01 0.06460017-2i
## 2009-03-01 0.78063172+0i
## 2009-04-01 0.25625164-2i
## 2009-05-01 0.58594426+0i
## 2009-06-01 0.59209478-2i
```

```
Im(x + u)
```

```
##            X.1
## 2009-01-01   0
## 2009-02-01   2
## 2009-03-01   0
## 2009-04-01   2
## 2009-05-01   0
## 2009-06-01   2
```

```
Mod(x + u)
```

```
##                  X.1
## 2009-01-01 0.7819735
## 2009-02-01 2.0010430
## 2009-03-01 0.7806317
## 2009-04-01 2.0163494
## 2009-05-01 0.5859443
## 2009-06-01 2.0858035
```

```
Re(x + u)
```

```
##                   X.1
## 2009-01-01 0.78197354
## 2009-02-01 0.06460017
## 2009-03-01 0.78063172
## 2009-04-01 0.25625164
## 2009-05-01 0.58594426
## 2009-06-01 0.59209478
```

timeSeries:

```
Arg(s + u)
```

```
## GMT
##                  S.1
## 2009-01-01 0.000000
## 2009-02-01 1.538507
## 2009-03-01 0.000000
## 2009-04-01 1.443365
## 2009-05-01 0.000000
## 2009-06-01 1.282970
```

```
Conj(s + u)
```

```
## GMT
##                        S.1
## 2009-01-01 0.78197354+0i
## 2009-02-01 0.06460017-2i
## 2009-03-01 0.78063172+0i
## 2009-04-01 0.25625164-2i
## 2009-05-01 0.58594426+0i
## 2009-06-01 0.59209478-2i
```

```
Im(s + u)
```

```
## GMT
##             S.1
## 2009-01-01   0
## 2009-02-01   2
## 2009-03-01   0
## 2009-04-01   2
## 2009-05-01   0
## 2009-06-01   2
```

```
Mod(s + u)
```

```
## GMT
##                   S.1
## 2009-01-01 0.7819735
## 2009-02-01 2.0010430
## 2009-03-01 0.7806317
## 2009-04-01 2.0163494
## 2009-05-01 0.5859443
## 2009-06-01 2.0858035
```

```
Re(s + u)
```

```
## GMT
##                    S.1
## 2009-01-01 0.78197354
## 2009-02-01 0.06460017
## 2009-03-01 0.78063172
## 2009-04-01 0.25625164
## 2009-05-01 0.58594426
## 2009-06-01 0.59209478
```

2.1.8 缺失值处理

需要加载的 R 包。

```
library(zoo)
library(xts)
library(timeSeries)
```

2.1.8.1 如何在单变量时间序列中忽略缺失值? 注意，这里的忽略有多种含义。其最基本的含义是将 NA 从数据集中删除。更广泛的还意味着对缺失值进行替代或者差值处理。先看其基本含义。

常规数据

```
data <- rnorm(9)
data[c(1, 7)] <- NA
data
```

```
## [1]          NA  1.24516477  0.34790446
## [4] -0.33765469 -0.37774291 -0.42397850
## [7]          NA  0.02463554 -1.71410895
```

```
charvec <- paste("2009-0", 1:9, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-07-01" "2009-08-01" "2009-09-01"
```

zoo:

```
z <- zoo(data, as.Date(charvec))
na.omit(z)
```

```
##  2009-02-01  2009-03-01  2009-04-01  2009-05-01
##  1.24516477  0.34790446 -0.33765469 -0.37774291
##  2009-06-01  2009-08-01  2009-09-01
## -0.42397850  0.02463554 -1.71410895
```

xts:

```
x <- xts(data, as.Date(charvec))
na.omit(x)
```

```
##                    [,1]
## 2009-02-01  1.24516477
```

```
## 2009-03-01  0.34790446
## 2009-04-01 -0.33765469
## 2009-05-01 -0.37774291
## 2009-06-01 -0.42397850
## 2009-08-01  0.02463554
## 2009-09-01 -1.71410895
```

timeSeries:

```
s <- timeSeries(data, charvec)
na.omit(s)
```

```
## GMT
##                    TS.1
## 2009-02-01  1.24516477
## 2009-03-01  0.34790446
## 2009-04-01 -0.33765469
## 2009-05-01 -0.37774291
## 2009-06-01 -0.42397850
## 2009-08-01  0.02463554
## 2009-09-01 -1.71410895
```

2.1.8.2 如何在多变量时间序列中忽略缺失值? 常规数据

```
data <- matrix(rnorm(7 * 3), ncol = 3)
data[1, 1] <- NA
data[3, 1:2] <- NA
data[4, 2] <- NA
data
```

```
##             [,1]       [,2]          [,3]
## [1,]          NA  0.9394643 -0.0811156157
## [2,]  0.8298238  1.3002218  0.8310896735
## [3,]          NA         NA  0.0843383659
```

```
## [4,] -1.4033585         NA  0.8276117578
## [5,] -0.5375709  0.7683343 -1.8330234942
## [6,] -0.9351130  1.0228431  1.7274147906
## [7,]  0.1956922 -0.5792542  0.0005245919
```

```
charvec <- paste("2009-0", 1:7, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-07-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
na.omit(Z)
```

```
##
## 2009-02-01  0.8298238  1.3002218  0.8310896735
## 2009-05-01 -0.5375709  0.7683343 -1.8330234942
## 2009-06-01 -0.9351130  1.0228431  1.7274147906
## 2009-07-01  0.1956922 -0.5792542  0.0005245919
## attr(,"na.action")
## [1] 1 3 4
## attr(,"class")
## [1] omit
```

xts:

```
X <- xts(data, as.Date(charvec))
na.omit(X)
```

```
##                   [,1]       [,2]          [,3]
## 2009-02-01  0.8298238  1.3002218  0.8310896735
```

```
## 2009-05-01 -0.5375709  0.7683343 -1.8330234942
## 2009-06-01 -0.9351130  1.0228431  1.7274147906
## 2009-07-01  0.1956922 -0.5792542  0.0005245919
```

timeSeries:

```
s <- timeSeries(data, charvec)
na.omit(s)
```

```
## GMT
##                   TS.1       TS.2          TS.3
## 2009-02-01  0.8298238  1.3002218  0.8310896735
## 2009-05-01 -0.5375709  0.7683343 -1.8330234942
## 2009-06-01 -0.9351130  1.0228431  1.7274147906
## 2009-07-01  0.1956922 -0.5792542  0.0005245919
```

2.1.8.3 如何在一个多变量时间序列对象中, 使用 0, 均值或中位值等来替换缺失值? 常规数据

```
data <- rnorm(9)
data[c(1, 7)] <- NA
charvec <- paste("2009-0", 1:9, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-07-01" "2009-08-01" "2009-09-01"
```

处理金融时间序列数据时, 人们会用一个常值, 比如 0, 均值或者中位值来替换缺失值。

zoo:

用 0 来取代一个金融收益序列的缺失值。

```
z <- zoo(data, as.Date(charvec))
z[is.na(z)] <- 0
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.0000000  0.4632707  1.6475984  0.7720409
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
## -0.0160609  0.0055304  0.0000000  1.0962222
## 2009-09-01
##  0.4628609
```

或者用样本均值来替换一个金融收益序列的缺失值。

```
z <- zoo(data, as.Date(charvec))
z[is.na(z)] <- mean(z, na.rm = TRUE)
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.6330661  0.4632707  1.6475984  0.7720409
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
## -0.0160609  0.0055304  0.6330661  1.0962222
## 2009-09-01
##  0.4628609
```

或者用一个均值的稳健估计, 即中位数, 来替换一个金融收益序列的缺失值。

```
z <- zoo(data, as.Date(charvec))
z[is.na(z)] <- median(z, na.rm = TRUE)
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.4632707  0.4632707  1.6475984  0.7720409
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
## -0.0160609  0.0055304  0.4632707  1.0962222
```

```
## 2009-09-01
##  0.4628609
```

xts:

```
x <- xts(data, as.Date(charvec))
x[is.na(x)] <- 0
x
```

```
##                  [,1]
## 2009-01-01  0.0000000
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.0000000
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

```
x <- xts(data, as.Date(charvec))
x[is.na(x)] <- mean(x, na.rm = TRUE)
x
```

```
##                  [,1]
## 2009-01-01  0.6330661
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.6330661
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

```
x <- xts(data, as.Date(charvec))
x[is.na(x)] <- median(x, na.rm = TRUE)
x
```

```
##                   [,1]
## 2009-01-01  0.4632707
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.4632707
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

timeSeries:

```
s <- timeSeries(data, charvec)
s[is.na(s)] <- 0
s
```

```
## GMT
##                   TS.1
## 2009-01-01  0.0000000
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.0000000
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

```
s <- timeSeries(data, charvec)
s[is.na(s)] <- mean(s, na.rm = TRUE)
s
```

```
## GMT
##                   TS.1
## 2009-01-01  0.6330661
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.6330661
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

```
s <- timeSeries(data, charvec)
s[is.na(s)] <- median(s, na.rm = TRUE)
s
```

```
## GMT
##                   TS.1
## 2009-01-01  0.4632707
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.4632707
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

2.1.8.4 如何替换多变量时间序列对象中的缺失值? 处理金融时间序列数据时, 人们会用一个常值, 比如 0, 均值或者中位值来替换缺失值。

常规数据

```
data <- matrix(rnorm(7 * 3), ncol = 3)
data[1, 1] <- NA
data[3, 1:2] <- NA
data[4, 2] <- NA
data
```

```
##             [,1]       [,2]       [,3]
## [1,]          NA -0.7271600  0.0628340
## [2,] -1.36159100  0.4917101  0.6010447
## [3,]          NA         NA  0.2461066
## [4,]  1.04683756         NA  0.2281587
## [5,] -0.23731493  1.6683801 -0.3504648
## [6,] -0.04646330 -0.2019813 -1.0855506
## [7,]  0.02165426 -1.0717262  1.6455858
```

```
charvec <- paste("2009-0", 1:7, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-07-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
Z
```

```
##
## 2009-01-01          NA -0.7271600  0.0628340
## 2009-02-01 -1.36159100  0.4917101  0.6010447
## 2009-03-01          NA         NA  0.2461066
## 2009-04-01  1.04683756         NA  0.2281587
```

```
## 2009-05-01 -0.23731493  1.6683801 -0.3504648
## 2009-06-01 -0.04646330 -0.2019813 -1.0855506
## 2009-07-01  0.02165426 -1.0717262  1.6455858
```

```
Z[is.na(Z)] <- 0
Z
```

```
##
## 2009-01-01  0.00000000 -0.7271600  0.0628340
## 2009-02-01 -1.36159100  0.4917101  0.6010447
## 2009-03-01  0.00000000          NA  0.2461066
## 2009-04-01  1.04683756          NA  0.2281587
## 2009-05-01 -0.23731493  1.6683801 -0.3504648
## 2009-06-01 -0.04646330 -0.2019813 -1.0855506
## 2009-07-01  0.02165426 -1.0717262  1.6455858
```

xts:

```
X <- xts(data, as.Date(charvec))
X
```

```
##                   [,1]       [,2]       [,3]
## 2009-01-01          NA -0.7271600  0.0628340
## 2009-02-01 -1.36159100  0.4917101  0.6010447
## 2009-03-01          NA         NA  0.2461066
## 2009-04-01  1.04683756         NA  0.2281587
## 2009-05-01 -0.23731493  1.6683801 -0.3504648
## 2009-06-01 -0.04646330 -0.2019813 -1.0855506
## 2009-07-01  0.02165426 -1.0717262  1.6455858
```

```
X[is.na(X)] <- 0
X
```

```
##                   [,1]       [,2]       [,3]
```

```
## 2009-01-01  0.00000000 -0.7271600  0.0628340
## 2009-02-01 -1.36159100  0.4917101  0.6010447
## 2009-03-01  0.00000000  0.0000000  0.2461066
## 2009-04-01  1.04683756  0.0000000  0.2281587
## 2009-05-01 -0.23731493  1.6683801 -0.3504648
## 2009-06-01 -0.04646330 -0.2019813 -1.0855506
## 2009-07-01  0.02165426 -1.0717262  1.6455858
```

timeSeries:

```
S <- timeSeries(data, as.Date(charvec))
S
```

```
## GMT
##                   TS.1       TS.2       TS.3
## 2009-01-01          NA -0.7271600  0.0628340
## 2009-02-01 -1.36159100  0.4917101  0.6010447
## 2009-03-01          NA         NA  0.2461066
## 2009-04-01  1.04683756         NA  0.2281587
## 2009-05-01 -0.23731493  1.6683801 -0.3504648
## 2009-06-01 -0.04646330 -0.2019813 -1.0855506
## 2009-07-01  0.02165426 -1.0717262  1.6455858
```

```
S[is.na(S)] <- 0
S
```

```
## GMT
##                   TS.1       TS.2       TS.3
## 2009-01-01  0.00000000 -0.7271600  0.0628340
## 2009-02-01 -1.36159100  0.4917101  0.6010447
## 2009-03-01  0.00000000  0.0000000  0.2461066
## 2009-04-01  1.04683756  0.0000000  0.2281587
## 2009-05-01 -0.23731493  1.6683801 -0.3504648
## 2009-06-01 -0.04646330 -0.2019813 -1.0855506
## 2009-07-01  0.02165426 -1.0717262  1.6455858
```

2.1.8.5 如何对单变量时间序列对象中的缺失值进行插值处理? 在 zoo 包和 xts 包中, na.approx 和 na.spline 函数能完成这样的操作。timeSeries 对象可以通过选择 na.omit 函数中的 method 参数完成缺失值的插值操作。

常规数据

```
data <- 1:9
data[c(1, 7)] <- NA
data
```

```
## [1] NA  2  3  4  5  6 NA  8  9
```

```
charvec <- paste("2009-0", 1:9, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-07-01" "2009-08-01" "2009-09-01"
```

zoo:

使用 approx() 函数进行线性插值。

```
z <- zoo(data, as.Date(charvec))
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##         NA          2          3          4
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
##          5          6         NA          8
## 2009-09-01
##          9
```

```
na.approx(z)
```

```
## 2009-02-01 2009-03-01 2009-04-01 2009-05-01
##   2.000000   3.000000   4.000000   5.000000
## 2009-06-01 2009-07-01 2009-08-01 2009-09-01
##   6.000000   6.983607   8.000000   9.000000
```

使用 spline 函数进行样条插值。

```
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##         NA          2          3          4
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
##          5          6         NA          8
## 2009-09-01
##          9
```

```
na.spline(z)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.6305683  2.0000000  3.0000000  4.0000000
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
##  5.0000000  6.0000000  6.9796640  8.0000000
## 2009-09-01
##  9.0000000
```

末次观测值结转法

```
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##         NA          2          3          4
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
##          5          6         NA          8
## 2009-09-01
##          9
```

```
na.locf(z)
```

```
## 2009-02-01 2009-03-01 2009-04-01 2009-05-01
##          2          3          4          5
## 2009-06-01 2009-07-01 2009-08-01 2009-09-01
##          6          6          8          9
```

裁掉首尾缺失值。

```
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##         NA          2          3          4
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
##          5          6         NA          8
## 2009-09-01
##          9
```

```
na.trim(z, sides = "left")
```

```
## 2009-02-01 2009-03-01 2009-04-01 2009-05-01
##          2          3          4          5
## 2009-06-01 2009-07-01 2009-08-01 2009-09-01
##          6         NA          8          9
```

xts:

```
xts <- xts(data, as.Date(charvec))
xts
```

```
##            [,1]
## 2009-01-01   NA
## 2009-02-01    2
## 2009-03-01    3
```

```
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
## 2009-07-01   NA
## 2009-08-01    8
## 2009-09-01    9
```

```
na.approx(xts)
```

```
##                 [,1]
## 2009-02-01 2.000000
## 2009-03-01 3.000000
## 2009-04-01 4.000000
## 2009-05-01 5.000000
## 2009-06-01 6.000000
## 2009-07-01 6.983607
## 2009-08-01 8.000000
## 2009-09-01 9.000000
```

```
x
```

```
##                   [,1]
## 2009-01-01  0.4632707
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.4632707
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

```
na.spline(x)
```

```
##                   [,1]
## 2009-01-01  0.4632707
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.4632707
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

末次观测值结转法

```
x
```

```
##                   [,1]
## 2009-01-01  0.4632707
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.4632707
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

```
na.locf(x)
```

```
##                   [,1]
## 2009-01-01  0.4632707
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
```

```
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.4632707
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

裁掉首尾缺失值

```
x
```

```
##                  [,1]
## 2009-01-01  0.4632707
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.4632707
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

```
na.trim(x, sides = "left")
```

```
##                  [,1]
## 2009-01-01  0.4632707
## 2009-02-01  0.4632707
## 2009-03-01  1.6475984
## 2009-04-01  0.7720409
## 2009-05-01 -0.0160609
## 2009-06-01  0.0055304
## 2009-07-01  0.4632707
## 2009-08-01  1.0962222
## 2009-09-01  0.4628609
```

timeSeries:

timeSeries 对象的插值处理可以由函数 approx 处理，该函数采用的是线性插值方法。在时间序列的首末位置对 NA 进行插值或者移除操作。

```
s <- timeSeries(data, charvec)
na.omit(s, "ir")
```

```
## GMT
##            TS.1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
## 2009-07-01    6
## 2009-08-01    8
## 2009-09-01    9
```

用 0 替换，时间序列首末位置的缺失值。

```
s <- timeSeries(data, charvec)
na.omit(s, "iz")
```

```
## GMT
##            TS.1
## 2009-01-01    0
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
## 2009-07-01    6
## 2009-08-01    8
## 2009-09-01    9
```

对时间序列首末位置的缺失值进行插值和外推处理。

```
s <- timeSeries(data, charvec)
na.omit(s, "ie")
```

```
## GMT
##            TS.1
## 2009-01-01    2
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
## 2009-07-01    6
## 2009-08-01    8
## 2009-09-01    9
```

通过设置参数 interp=c("before", "linear", "after")，可以选择插值方式。before 选项，末次观测值结转法，after 选项，下次观测结转法。

```
sn <- na.omit(s, method = "iz", interp = "before")
sn[is.na(s)]
```

```
## [1] 0 6
```

```
sn <- na.omit(s, method = "iz", interp = "linear")
sn[is.na(s)]
```

```
## [1] 0 7
```

```
sn <- na.omit(s, method = "iz", interp = "after")
sn[is.na(s)]
```

```
## [1] 0 8
```

请注意：通过设定 approx 函数中的默认选项可以调整相应的线性插值操作。

```
args(approx)
```

```
## function (x, y = NULL, xout, method = "linear", n = 50, yleft,
##     yright, rule = 1, f = 0, ties = mean, na.rm = TRUE)
## NULL
```

可以对一个单变量时间序列对象应用 na.contiguous 函数吗?

na.contiguous 函数将返回时间序列对象中最长的那个连续无缺失值的序列片段，如果对象中有两个等长的序列片段，函数将返回最前面那个。

常规数据

```
data <- rnorm(12)
data[c(3, 8)] = NA
data
```

```
##  [1] -0.9824329 -0.7664704         NA  0.8327554
##  [5] -0.5733652  1.4879523 -0.6665595         NA
##  [9] -0.6109797  1.4891481 -0.9284593  0.7051616
```

```
charvec <- as.character(timeCalendar(2009))
charvec
```

```
##  [1] "2009-01-01" "2009-02-01" "2009-03-01"
##  [4] "2009-04-01" "2009-05-01" "2009-06-01"
##  [7] "2009-07-01" "2009-08-01" "2009-09-01"
## [10] "2009-10-01" "2009-11-01" "2009-12-01"
```

ts:

```
t <- ts(data, start = 2009, frequency = 12)
t
```

```
##              Jan        Feb        Mar        Apr
## 2009 -0.9824329 -0.7664704         NA  0.8327554
##              May        Jun        Jul        Aug
## 2009 -0.5733652  1.4879523 -0.6665595         NA
##              Sep        Oct        Nov        Dec
## 2009 -0.6109797  1.4891481 -0.9284593  0.7051616
```

```
na.contiguous(t)
```

```
##              Apr        May        Jun        Jul
## 2009  0.8327554 -0.5733652  1.4879523 -0.6665595
```

zoo:

```
z <- zoo(data, as.Date(charvec))
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
## -0.9824329 -0.7664704         NA  0.8327554
## 2009-05-01 2009-06-01 2009-07-01 2009-08-01
## -0.5733652  1.4879523 -0.6665595         NA
## 2009-09-01 2009-10-01 2009-11-01 2009-12-01
## -0.6109797  1.4891481 -0.9284593  0.7051616
```

```
na.contiguous(z)
```

```
## 2009-04-01 2009-05-01 2009-06-01 2009-07-01
##  0.8327554 -0.5733652  1.4879523 -0.6665595
```

xts:

```
x <- xts(data, as.Date(charvec))
x
```

```
##                  [,1]
## 2009-01-01 -0.9824329
## 2009-02-01 -0.7664704
## 2009-03-01         NA
## 2009-04-01  0.8327554
## 2009-05-01 -0.5733652
## 2009-06-01  1.4879523
## 2009-07-01 -0.6665595
## 2009-08-01         NA
## 2009-09-01 -0.6109797
## 2009-10-01  1.4891481
## 2009-11-01 -0.9284593
## 2009-12-01  0.7051616
```

```
na.contiguous(x)
```

```
##                  [,1]
## 2009-04-01  0.8327554
## 2009-05-01 -0.5733652
## 2009-06-01  1.4879523
## 2009-07-01 -0.6665595
```

timeSeries:

```
s <- timeSeries(data, charvec)
s
```

```
## GMT
##                  TS.1
## 2009-01-01 -0.9824329
## 2009-02-01 -0.7664704
## 2009-03-01         NA
## 2009-04-01  0.8327554
## 2009-05-01 -0.5733652
```

```
## 2009-06-01  1.4879523
## 2009-07-01 -0.6665595
## 2009-08-01         NA
## 2009-09-01 -0.6109797
## 2009-10-01  1.4891481
## 2009-11-01 -0.9284593
## 2009-12-01  0.7051616
```

```
na.contiguous(s)
```

```
## GMT
##                   TS.1
## 2009-04-01  0.8327554
## 2009-05-01 -0.5733652
## 2009-06-01  1.4879523
## 2009-07-01 -0.6665595
```

2.1.9 时间序列对象的查看与绘图

2.1.9.1 时间序列对象的显示 需加载的 R 包

```
library(zoo)
library(xts)
library(timeSeries)
```

2.1.9.2 如何查看时间序列对象？

时间序列的对象在 R 中的显示方式与其它类型数据的显示一样，键入数据名可直接显示。

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

xts:

```
zoo(data, as.Date(charvec))
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          4
## 2009-05-01 2009-06-01
##          5          6
```

xts:

```
xts(data, as.Date(charvec))
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

timeSeries:

```
timeSeries(data, charvec)
```

```
## GMT
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

当然，可以用 print 函数在屏幕上打印时间序列对象。

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
print(zoo(data, as.Date(charvec)))
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          4
## 2009-05-01 2009-06-01
##          5          6
```

xts:

```
print(xts(data, as.Date(charvec)))
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

timeSeries:

```
print(timeSeries(data, charvec))
```

```
## GMT
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

鉴于 timeSeries 对象属于 S4 类，因此也可以用 show 函数来打印 timeSeries 对象。

```
show(timeSeries(data, charvec))
```

```
## GMT
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

2.1.9.3 如何控制单变量时间序列对象的显示形式? 常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

使用 print 函数打印单变量 zoo 类型时间序列对象时，数据一般以水平格式显示。

```
print(zoo(data, as.Date(charvec)))
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          4
## 2009-05-01 2009-06-01
##          5          6
```

xts:

用 print 函数打印单变量 xts 类型时间序列对象时，数据一般以竖直格式显示。

```
print(xts(data, as.Date(charvec)))
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

当然，也可以将 zoo 对象以竖直格式显示或者将 xts 对象以水平格式显示。

```
print(as.zoo(xts(data, as.Date(charvec))))
```

```
##
## 2009-01-01 1
## 2009-02-01 2
## 2009-03-01 3
```

```
## 2009-04-01 4
## 2009-05-01 5
## 2009-06-01 6
```

```
print(as.xts(xts(data, as.Date(charvec))))
```

```
##            [,1]
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

timeSeries:

用 print 函数打印 timeSeries 对象时，print 函数提供了 style 参数来调整 timeSeries 对象的显示格式。对于单变量 timeSeries 类型的时间序列数据，print 的默认为竖直显示。

```
s <- timeSeries(data, charvec)
print(s)
```

```
## GMT
##            TS.1
## 2009-01-01    1
## 2009-02-01    2
## 2009-03-01    3
## 2009-04-01    4
## 2009-05-01    5
## 2009-06-01    6
```

当然，也可以以水平格式显示。

```
print(s, style = "h")
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##          1          2          3          4
## 2009-05-01 2009-06-01
##          5          6
```

2.1.9.4 除了用 ISO（国际标准格式）日期/时间格式之外，还能用其它格式来显示时间序列对象吗? 常规数据

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

目前没有。

xts:

目前没有。

timeSeries:

用 print 函数打印 timeSeries 对象时，print 函数提供了一个 format 参数来调整 timeSeries 对象的显示类型。

```
s <- timeSeries(pi, "2009-05-08 19:26:22", units = "s")
print(s)
```

```
## GMT
##                             s
## 2009-05-08 19:26:22 3.141593
```

```
print(s, format = "%m/%d/%y %H:%M")
```

```
## GMT
##                         s
## 05/08/09 19:26 3.141593
```

```
print(s, format = "%m/%d/%y %A")
```

```
## GMT
##                          s
## 05/08/09 星期五 3.141593
```

```
print(s, format = "DayNo %j Hour: %H")
```

```
## GMT
##                             s
## DayNo 128 Hour: 19 3.141593
```

第一个例子使用的是默认 ISO 格式，第二例子是包含星期全名的 month-day-year 格式，而最后一个例子展示了此日期在一年中的位置，且展示了具体时间中的小时。时间序列对象能按照 R 中的 ts 格式显示吗?

zoo:

zoo 类型的等间隔时间序列对象可以以 ts 格式显示，非等间隔不能。

xts:

目前没有。

timeSeries:

print 函数打印 timeSeries 对象时提供的 style 参数里面有 ts 选项。例如，对于一个月度记录的时间序列数据:

```
data <- 1:6
charvec <- paste("2009-0", 1:6, "-01", sep = "")
s <- timeSeries(data, charvec)
print(s, style = "ts")
```

```
##      Jan Feb Mar Apr May Jun
## 2009   1   2   3   4   5   6
```

对于一个季度记录的时间序列数据

```
data <- 1:4
charvec <- paste("2009-", c("03", "06", "09", "12"), "-01", sep = "")
s <- timeSeries(data, charvec)
print(s, style = "ts", by = "quarter")
```

```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 2009                1    2
## 2010    3    4    1    2
## 2011    3    4    1    2
```

2.1.9.5 如何调整时间序列对象的时区设置? zoo:

目前，无法直接做到。

xts:

目前，无法直接做到。

timeSeries:

print 函数用来打印 timeSeries 对象时提供了一个 FinCenter 参数来设置时间序列对象的时区信息。针对一个月度时间序列数据:

```
data <- rnorm(6)
charvec <- paste("2009-0", 1:6, "-01 16:00:00", sep = "")
s <- timeSeries(data, charvec, zone = "Chicago", FinCenter = "Zurich")
print(s, FinCenter = "Chicago")
```

```
## Chicago
##                            TS.1
## 2009-01-01 16:00:00  1.3565302
## 2009-02-01 16:00:00 -1.2217405
## 2009-03-01 16:00:00  1.2766963
## 2009-04-01 16:00:00  0.1316002
## 2009-05-01 16:00:00 -0.6394686
## 2009-06-01 16:00:00  1.4848902
```

```
print(s, FinCenter = "Zurich")
```

```
## Zurich
##                            TS.1
## 2009-01-01 23:00:00  1.3565302
## 2009-02-01 23:00:00 -1.2217405
## 2009-03-01 23:00:00  1.2766963
## 2009-04-01 23:00:00  0.1316002
## 2009-05-01 23:00:00 -0.6394686
## 2009-06-01 23:00:00  1.4848902
```

```
print(s, FinCenter = "Tokyo")
```

```
## Tokyo
##                            TS.1
## 2009-01-02 07:00:00  1.3565302
## 2009-02-02 07:00:00 -1.2217405
## 2009-03-02 07:00:00  1.2766963
## 2009-04-02 06:00:00  0.1316002
## 2009-05-02 06:00:00 -0.6394686
## 2009-06-02 06:00:00  1.4848902
```

2.1.10 时间序列对象的图形展示

需要加载的 R 包

```
library(zoo)
library(xts)
library(timeSeries)
```

2.1.10.1 如何用 plot 函数绘制单变量时间序列数据? 常规数据

```
set.seed(1953)
data <- runif(6)
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

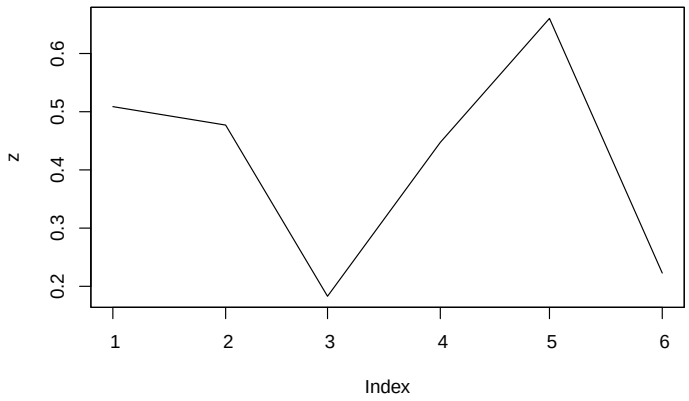
```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
args(plot.zoo)
```

```
## function (x, y = NULL, screens, plot.type, panel = lines, xlab = "Index",
##     ylab = NULL, main = NULL, xlim = NULL, ylim = NULL, xy.labels = FALSE,
##     xy.lines = NULL, yax.flip = FALSE, oma = c(6, 0, 5, 0), mar = c(0,
##         5.1, 0, if (yax.flip) 5.1 else 2.1), col = 1, lty = 1,
##     lwd = 1, pch = 1, type = "l", log = "", nc, widths = 1, heights = 1,
##     ...)
## NULL
```

```
z <- zoo(data, as.POSIXct(charvec, tz = "GMT"))
plot(z)
```



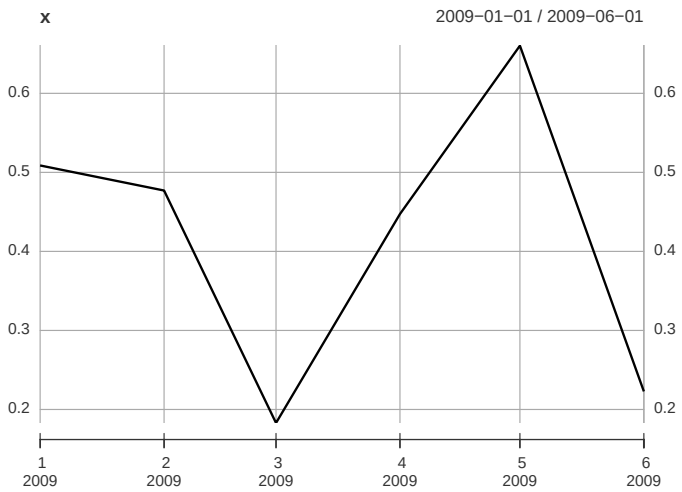
注意 X 轴的刻度上没有标明年份信息。

xts:

```
args(plot.xts)
```

```
## function (x, y = NULL, ..., subset = "", panels = NULL, multi.panel = FALSE,
##     col = 1:8, up.col = NULL, dn.col = NULL, bg = "#FFFFFF",
##     type = "l", lty = 1, lwd = 2, lend = 1, main = deparse(substitute(x)),
##     main.timespan = TRUE, observation.based = FALSE, log = FALSE,
##     ylim = NULL, yaxis.same = TRUE, yaxis.left = TRUE, yaxis.right = TRUE,
##     yaxis.ticks = 5, major.ticks = "auto", minor.ticks = NULL,
##     grid.ticks.on = "auto", grid.ticks.lwd = 1, grid.ticks.lty = 1,
##     grid.col = "darkgray", labels.col = "#333333", format.labels = TRUE,
##     grid2 = "#F5F5F5", legend.loc = NULL, extend.xaxis = FALSE)
## NULL
```

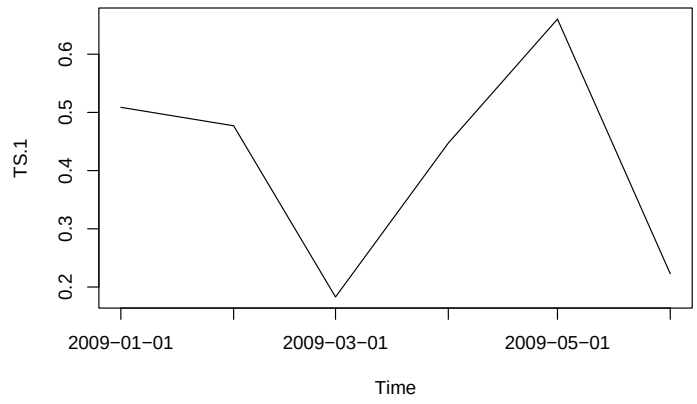
```
x <- xts(data, as.POSIXct(charvec, tz = "GMT"))
plot(x)
```



timeSeries:

timeSeries 对象可以用 S4 类型的绘图方法来绘制。

```
s <- timeSeries(data, charvec)
plot(s)
```



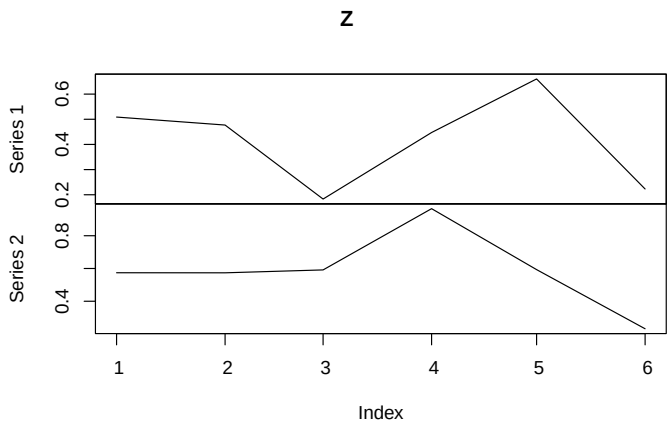
2.1.10.2 如何用 plot 函数绘制多变量时间序列数据? 常规数据

```
set.seed(1953)
data <- matrix(runif(12), ncol = 2)
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.POSIXct(charvec, tz = "GMT"))
plot(Z)
```



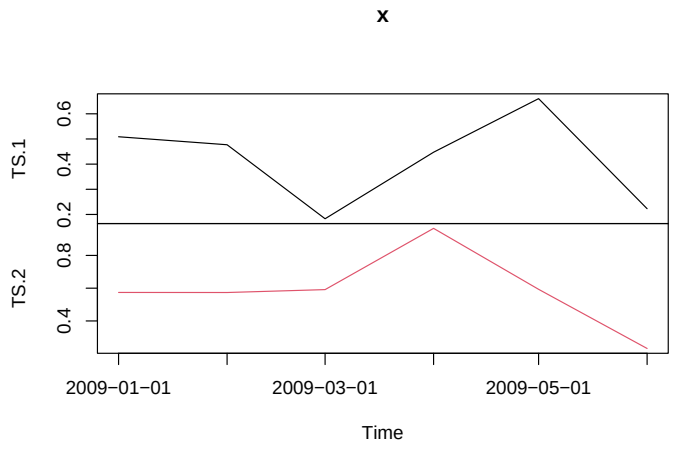
xts:

```
X <- xts(data, as.POSIXct(charvec, tz = "GMT"))
plot(X)
```

xts 对象不支持将多变量时间序列数据绘制在一张图形上。

timeSeries:

```
S <- timeSeries(data, charvec)
plot(S)
```



2.1.10.3 如何用 plot 函数将多变量时间序列数据绘制在同一张图上?

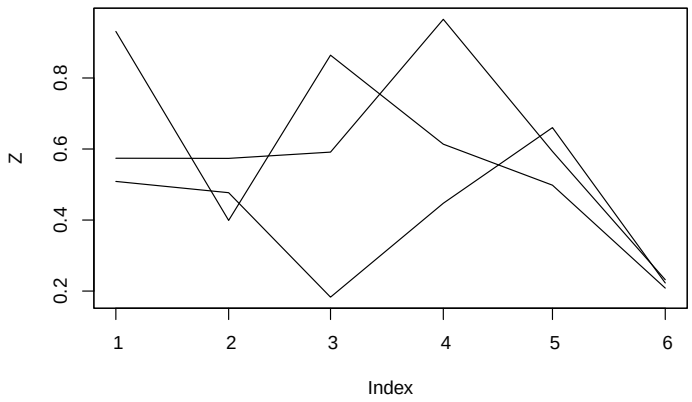
常规数据

```
set.seed(1953)
data <- matrix(runif(18), ncol = 3)
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.POSIXct(charvec, tz = "GMT"))
plot(Z, plot.type = "single")
```

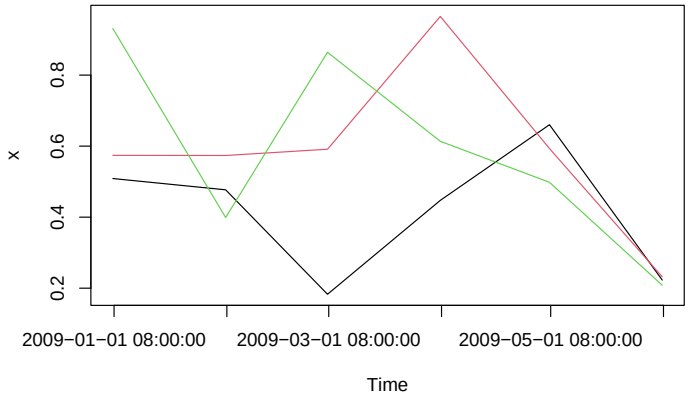


xts:

xts 对象不支持将多变量时间序列数据绘制在一张图形上。

timeSeries:

```
S <- timeSeries(data, as.POSIXct(charvec, FinCenter = "GMT"))
plot(S, plot.type = "single")
```



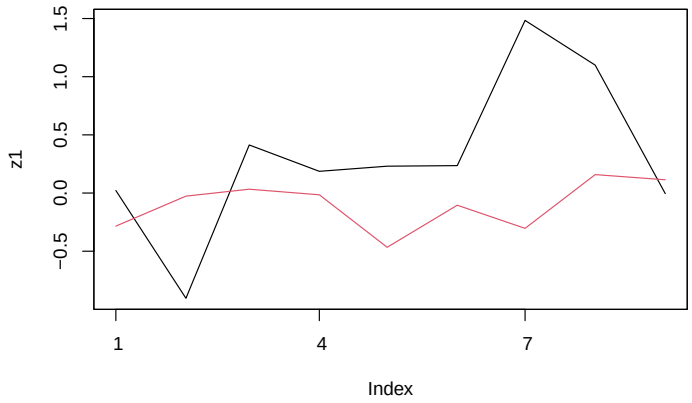
2.1.10.4 如何在已有图形上添加直线? 常规数据

```
set.seed(1953)
data1 <- rnorm(9)
data2 <- rnorm(9, sd = 0.2)
charvec <- paste("2009-0", 1:9, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-07-01" "2009-08-01" "2009-09-01"
```

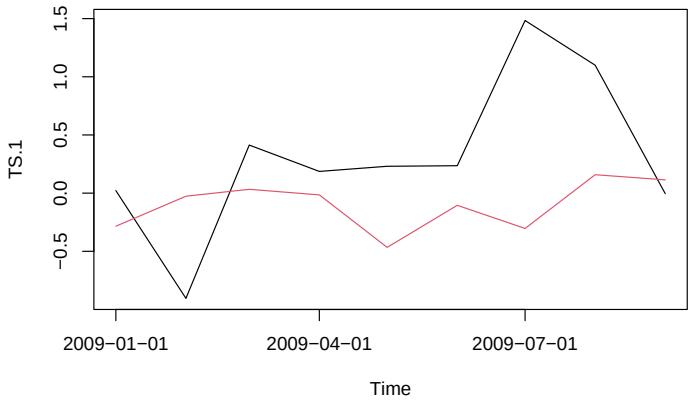
zoo:

```
z1 <- zoo(data1, as.POSIXct(charvec, tz = "GMT"))
z2 <- zoo(data2, as.POSIXct(charvec, tz = "GMT"))
plot(z1)
lines(z2, col = 2)
```



timeSeries:

```
s1 <- timeSeries(data1, charvec, FinCenter = "GMT")
s2 <- timeSeries(data2, charvec, FinCenter = "GMT")
plot(s1)
lines(s2, col = 2)
```



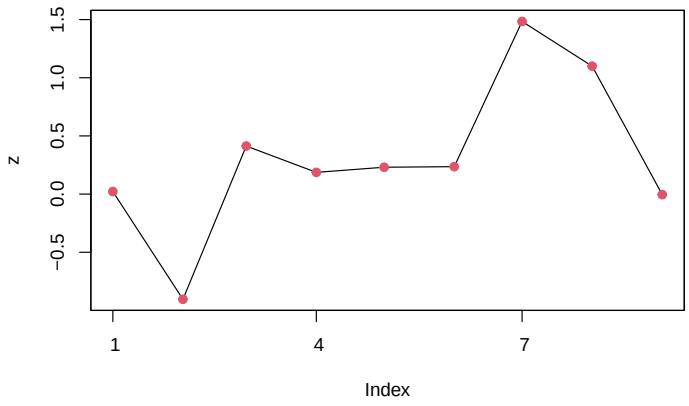
2.1.10.5 如何在现有图形中添加点? 常规数据

```
set.seed(1953)
data <- rnorm(9)
charvec <- paste("2009-0", 1:9, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-07-01" "2009-08-01" "2009-09-01"
```

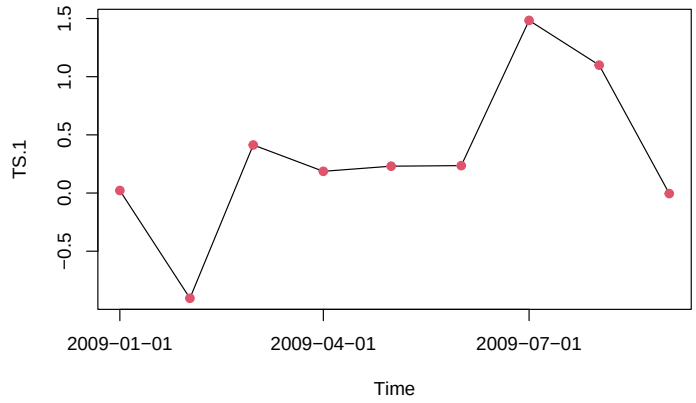
zoo:

```
z <- zoo(data, as.POSIXct(charvec, tz = "GMT"))
plot(z)
points(z, col = 2, pch = 19)
```



timeSeries:

```
s <- timeSeries(data, charvec, FinCenter = "GMT")
plot(s)
points(s, col = 2, pch = 19)
```



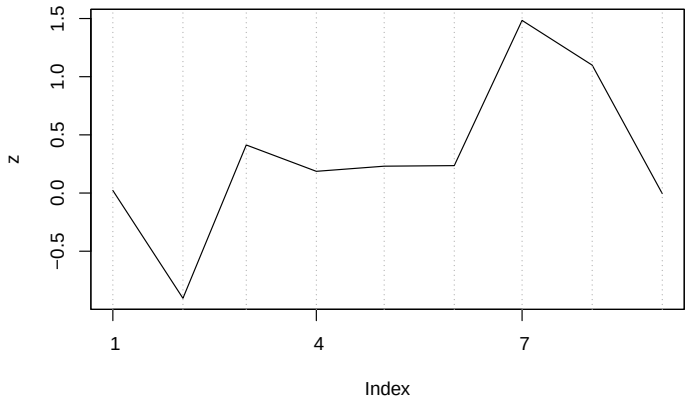
2.1.10.6 能在已有的时间序列图形上使用 abline 函数添加直线吗? 常规数据

```
set.seed(1953)
data <- rnorm(9)
charvec <- paste("2009-0", 1:9, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
## [7] "2009-07-01" "2009-08-01" "2009-09-01"
```

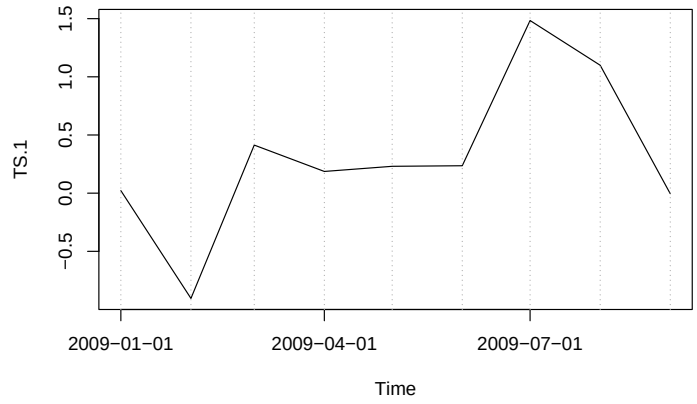
zoo:

```
z <- zoo(data, as.POSIXct(charvec, tz = "GMT"))
plot(z)
abline(v = index(z), col = "darkgrey", lty = 3)
```



timeSeries:

```
s <- timeSeries(data, charvec, FinCenter = "GMT")
plot(s)
abline(v = time(s), col = "darkgrey", lty = 3)
```



2.2 使用 R 函数

2.2.1 金融时间序列对象与 base 包中的函数

所需的 R 包

```
library(zoo)
library(xts)
library(timeSeries)
```

下面介绍几个与金融时间序列分析关系紧密的 R 函数，这些函数的分析对象即可以是多元时间序列也可以是单边量时间序列。

```
set.seed(1953)
data <- matrix(runif(18), ncol = 3)
charvec <- rev(paste("2009-0", 1:6, "-01", sep = ""))
charvec
```

```
## [1] "2009-06-01" "2009-05-01" "2009-04-01"
## [4] "2009-03-01" "2009-02-01" "2009-01-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
colnames(Z) = paste("Z", 1:3, sep = ".")
```

xts:

```
X <- xts(data, as.Date(charvec))
colnames(X) = paste("X", 1:3, sep = ".")
```

timeSeries:

```
S <- timeSeries(data, charvec)
colnames(S) = paste("S", 1:3, sep = ".")
```

zoo:

```
z <- Z[, 1]
```

xts:

```
x <- X[, 1]
```

timeSeries:

```
s <- S[, 1]
```

2.2.1.1 可以对金融时间序列对象应用 apply 函数吗？如何应用？

zoo:

```
apply(Z, 2, mean)
```

```
##       Z.1       Z.2       Z.3
## 0.4165161 0.5882129 0.5857227
```

xts:

```
apply(X, 2, mean)
```

```
##       X.1       X.2       X.3
## 0.4165161 0.5882129 0.5857227
```

timeSeries:

```
apply(S, 2, mean)
```

```
##       S.1       S.2       S.3
## 0.4165161 0.5882129 0.5857227
```

2.2.1.2 可以对金融时间序列对象应用 attach 函数吗? 应用 attach 函数之后，R 的数据库被纳入到工作区，如此一来，只需在 R 中键入数据集的名称，便可调用该数据集。attach 函数的对象只能是 lists,data frames 和 environments。因此，attach 函数针对金融时间序列对象不可用。

zoo:

```
print(try(attach(Z)))
```

xts:

```
print(try(attach(X)))
```

timeSeries:

```
attach(S)
colnames(S)
```

```
## [1] "S.1" "S.2" "S.3"
```

```
S
```

```
## GMT
##                  S.1       S.2       S.3
## 2009-06-01 0.5087459 0.5740212 0.9310609
## 2009-05-01 0.4770248 0.5736546 0.3990106
## 2009-04-01 0.1829114 0.5912718 0.8642168
## 2009-03-01 0.4472982 0.9652389 0.6135854
## 2009-02-01 0.6602838 0.5931596 0.4981308
## 2009-01-01 0.2228321 0.2319316 0.2083319
```

2.2.1.3 能对金融时间序列对象应用 diff 函数么? base 包中，diff 函数的作用是返回对象的差分序列。该函数可用于金融时间序列分析对象。

zoo:

```
diff(Z)
```

```
##                    Z.1           Z.2        Z.3
## 2009-02-01  0.43745169  0.3612279990  0.2897989
## 2009-03-01 -0.21298555  0.3720793007  0.1154546
## 2009-04-01 -0.26438683 -0.3739670936  0.2506314
## 2009-05-01  0.29411341 -0.0176171379 -0.4652063
## 2009-06-01  0.03172112  0.0003665467  0.5320503
```

```
diff(z)
```

```
##  2009-02-01  2009-03-01  2009-04-01  2009-05-01
##  0.43745169 -0.21298555 -0.26438683  0.29411341
##  2009-06-01
##  0.03172112
```

xts:

```
diff(X)
```

```
##                    X.1           X.2        X.3
## 2009-01-01          NA            NA         NA
## 2009-02-01  0.43745169  0.3612279990  0.2897989
## 2009-03-01 -0.21298555  0.3720793007  0.1154546
## 2009-04-01 -0.26438683 -0.3739670936  0.2506314
## 2009-05-01  0.29411341 -0.0176171379 -0.4652063
## 2009-06-01  0.03172112  0.0003665467  0.5320503
```

```
diff(x)
```

```
##                    X.1
## 2009-01-01          NA
## 2009-02-01  0.43745169
```

```
## 2009-03-01 -0.21298555
## 2009-04-01 -0.26438683
## 2009-05-01  0.29411341
## 2009-06-01  0.03172112
```

timeSeries:

```
diff(S)
```

```
## GMT
##                    S.1           S.2        S.3
## 2009-06-01          NA            NA         NA
## 2009-05-01 -0.03172112 -0.0003665467 -0.5320503
## 2009-04-01 -0.29411341  0.0176171379  0.4652063
## 2009-03-01  0.26438683  0.3739670936 -0.2506314
## 2009-02-01  0.21298555 -0.3720793007 -0.1154546
## 2009-01-01 -0.43745169 -0.3612279990 -0.2897989
```

```
diff(s)
```

```
## GMT
##                    S.1
## 2009-06-01          NA
## 2009-05-01 -0.03172112
## 2009-04-01 -0.29411341
## 2009-03-01  0.26438683
## 2009-02-01  0.21298555
## 2009-01-01 -0.43745169
```

2.2.1.4 能对金融时间序列对象应用 dim 函数么?

dim 函数用于返回对象的维度。该函数适用于金融时间序列对象。

zoo:

```
dim(Z)
```

```
## [1] 6 3
```

```
dim(z)
```

```
## NULL
```

xts:

```
dim(X)
```

```
## [1] 6 3
```

```
dim(x)
```

```
## [1] 6 1
```

timeSeries:

```
dim(S)
```

```
## [1] 6 3
```

```
dim(s)
```

```
## [1] 6 1
```

2.2.1.5 能对金融时间序列对象应用 rank 函数吗? rank 函数返回对象中每个元素的顺序位置。仅适用于 timeSeries 类型的金融时间序列对象。

zoo

```
print(try(rank(Z)))

print(try(rank(z)))
```

xts:

```
print(try(rank(X)))

print(try(rank(x)))
```

timeSeries:

```
rank(S)
```

```
## GMT
##            S.1 S.2 S.3
## 2009-06-01   5   3   6
## 2009-05-01   4   2   2
## 2009-04-01   1   4   5
## 2009-03-01   3   6   4
## 2009-02-01   6   5   3
## 2009-01-01   2   1   1
```

```
rank(s)
```

```
## GMT
##            S.1
## 2009-06-01   5
## 2009-05-01   4
## 2009-04-01   1
## 2009-03-01   3
## 2009-02-01   6
## 2009-01-01   2
```

2.2.1.6 能对金融时间序列对象应用 rev 函数么? rev 函数用于返回对象的逆排列。

zoo:

```
print(try(rev(Z)))
```

```
##                  Z.1       Z.2       Z.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
## 2009-04-01 0.4472982 0.9652389 0.6135854
## 2009-05-01 0.6602838 0.5931596 0.4981308
## 2009-06-01 0.2228321 0.2319316 0.2083319
```

```
rev(z)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.5087459  0.4770248  0.1829114  0.4472982
## 2009-05-01 2009-06-01
##  0.6602838  0.2228321
```

xts:

```
print(try(rev(X)))
```

```
##                  X.1       X.2       X.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
## 2009-04-01 0.4472982 0.9652389 0.6135854
## 2009-05-01 0.6602838 0.5931596 0.4981308
## 2009-06-01 0.2228321 0.2319316 0.2083319
```

```
rev(x)
```

```
##                  X.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
## 2009-04-01 0.4472982
## 2009-05-01 0.6602838
## 2009-06-01 0.2228321
```

timeSeries:

```
rev(S)
```

```
## GMT
##                  S.1       S.2       S.3
## 2009-01-01 0.2228321 0.2319316 0.2083319
## 2009-02-01 0.6602838 0.5931596 0.4981308
## 2009-03-01 0.4472982 0.9652389 0.6135854
## 2009-04-01 0.1829114 0.5912718 0.8642168
## 2009-05-01 0.4770248 0.5736546 0.3990106
## 2009-06-01 0.5087459 0.5740212 0.9310609
```

```
rev(s)
```

```
## GMT
##                  S.1
## 2009-01-01 0.2228321
## 2009-02-01 0.6602838
## 2009-03-01 0.4472982
## 2009-04-01 0.1829114
## 2009-05-01 0.4770248
## 2009-06-01 0.5087459
```

2.2.1.7 能对金融时间序列对象应用 sample 函数吗? sample 函数用于从一组数据中进行返回或者不返回抽样。

zoo:

```
print(try(sample(Z)))
```

```
## Error in rval[i, , drop = drop., ...] : 下标出界
## [1] "Error in rval[i, , drop = drop., ...] : 下标出界\n"
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <subscriptOutOfBoundsError in rval[i, , drop = drop., ...]: 下标出界>
```

```
sample(z)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.2228321  0.6602838  0.4472982  0.1829114
## 2009-05-01 2009-06-01
##  0.4770248  0.5087459
```

xts:

```
print(try(sample(X)))
```

```
## Error in `[.xts`(x, sample.int(length(x), size, replace, prob)) :
##   subscript out of bounds
## [1] "Error in `[.xts`(x, sample.int(length(x), size, replace, prob)) : \n  subscript
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in `[.xts`(x, sample.int(length(x), size, replace, prob)): subscript ou
```

```
sample(x)
```

```
##                  X.1
## 2009-01-01 0.2228321
## 2009-02-01 0.6602838
## 2009-03-01 0.4472982
## 2009-04-01 0.1829114
## 2009-05-01 0.4770248
## 2009-06-01 0.5087459
```

timeSeries:

```
sample(S)
```

```
## GMT
##                  S.1       S.2       S.3
## 2009-06-01 0.5087459 0.5740212 0.9310609
## 2009-03-01 0.4472982 0.9652389 0.6135854
## 2009-02-01 0.6602838 0.5931596 0.4981308
## 2009-01-01 0.2228321 0.2319316 0.2083319
## 2009-04-01 0.1829114 0.5912718 0.8642168
## 2009-05-01 0.4770248 0.5736546 0.3990106
```

```
sample(s)
```

```
## GMT
##                  S.1
## 2009-03-01 0.4472982
## 2009-06-01 0.5087459
## 2009-05-01 0.4770248
## 2009-02-01 0.6602838
## 2009-01-01 0.2228321
## 2009-04-01 0.1829114
```

2.2.1.8 能对金融时间序列对象应用 scale 函数吗？若能，其实如何操作的？ scale 函数的作用是将对象进行中心化/标准化。

zoo:

```
scale(Z)
```

```
##                   Z.1         Z.2        Z.3
## 2009-01-01 -1.0674168 -1.53452097 -1.3643678
## 2009-02-01  1.3434348  0.02130539 -0.3166682
## 2009-03-01  0.1696445  1.62386883  0.1007310
## 2009-04-01 -1.2874247  0.01317458  1.0068300
## 2009-05-01  0.3334715 -0.06270328 -0.6750139
## 2009-06-01  0.5082906 -0.06112455  1.2484889
## attr(,"scaled:center")
##       Z.1       Z.2       Z.3
## 0.4165161 0.5882129 0.5857227
## attr(,"scaled:scale")
##       Z.1       Z.2       Z.3
## 0.1814511 0.2321776 0.2766049
```

```
scale(z)
```

```
##
## 2009-01-01 -1.0674168
## 2009-02-01  1.3434348
## 2009-03-01  0.1696445
## 2009-04-01 -1.2874247
## 2009-05-01  0.3334715
## 2009-06-01  0.5082906
## attr(,"scaled:center")
## [1] 0.4165161
## attr(,"scaled:scale")
## [1] 0.1814511
```

xts:

```
scale(X)
```

```
##                   X.1         X.2        X.3
## 2009-01-01 -1.0674168 -1.53452097 -1.3643678
## 2009-02-01  1.3434348  0.02130539 -0.3166682
## 2009-03-01  0.1696445  1.62386883  0.1007310
## 2009-04-01 -1.2874247  0.01317458  1.0068300
## 2009-05-01  0.3334715 -0.06270328 -0.6750139
## 2009-06-01  0.5082906 -0.06112455  1.2484889
```

```
scale(x)
```

```
##                   X.1
## 2009-01-01 -1.0674168
## 2009-02-01  1.3434348
## 2009-03-01  0.1696445
## 2009-04-01 -1.2874247
## 2009-05-01  0.3334715
## 2009-06-01  0.5082906
```

timeSeries:

```
scale(S)
```

```
## GMT
##                   S.1         S.2        S.3
## 2009-06-01  0.5082906 -0.06112455  1.2484889
## 2009-05-01  0.3334715 -0.06270328 -0.6750139
## 2009-04-01 -1.2874247  0.01317458  1.0068300
## 2009-03-01  0.1696445  1.62386883  0.1007310
## 2009-02-01  1.3434348  0.02130539 -0.3166682
## 2009-01-01 -1.0674168 -1.53452097 -1.3643678
```

```
scale(s)
```

```
## GMT
##                   S.1
## 2009-06-01  0.5082906
## 2009-05-01  0.3334715
## 2009-04-01 -1.2874247
## 2009-03-01  0.1696445
## 2009-02-01  1.3434348
## 2009-01-01 -1.0674168
```

2.2.1.9 能对金融时间序列对象应用 sort 函数吗? sort 函数的作用是是将对象进行自然顺序排列或者逆序排列。具体可以参考 order 函数。

zoo:

```
print(try(sort(Z)))

sort(z)
```

xts:

```
print(try(sort(X)))
```

```
## Error in `[.xts`(x, order(x, na.last = na.last, decreasing = decreasing)) :
##   subscript out of bounds
## [1] "Error in `[.xts`(x, order(x, na.last = na.last, decreasing = decreasing)) : \n
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in `[.xts`(x, order(x, na.last = na.last, decreasing = decreasing)): su
```

```
sort(x)
```

```
##                  X.1
## 2009-01-01 0.2228321
## 2009-02-01 0.6602838
## 2009-03-01 0.4472982
## 2009-04-01 0.1829114
## 2009-05-01 0.4770248
## 2009-06-01 0.5087459
```

timeSeries:

```
sort(S)
```

```
## GMT
##                  S.1       S.2       S.3
## 2009-01-01 0.2228321 0.2319316 0.2083319
## 2009-02-01 0.6602838 0.5931596 0.4981308
## 2009-03-01 0.4472982 0.9652389 0.6135854
## 2009-04-01 0.1829114 0.5912718 0.8642168
## 2009-05-01 0.4770248 0.5736546 0.3990106
## 2009-06-01 0.5087459 0.5740212 0.9310609
```

```
sort(s)
```

```
## GMT
##                  S.1
## 2009-01-01 0.2228321
## 2009-02-01 0.6602838
## 2009-03-01 0.4472982
## 2009-04-01 0.1829114
## 2009-05-01 0.4770248
## 2009-06-01 0.5087459
```

2.2.1.10 能对金融时间序列对象应用 start 和 end 函数吗? start 函数和 end 函数用于返回对象的头部数据和尾部数据。针对金融时间序列对象时，其返回的是时间戳的头部部分和尾部部分。

zoo:

```
start(Z)
```

```
## [1] "2009-01-01"
```

```
end(Z)
```

```
## [1] "2009-06-01"
```

```
start(z)
```

```
## [1] "2009-01-01"
```

```
end(z)
```

```
## [1] "2009-06-01"
```

xts:

```
start(X)
```

```
## [1] "2009-01-01"
```

```
end(X)
```

```
## [1] "2009-06-01"
```

```
start(x)
```

```
## [1] "2009-01-01"
```

```
end(x)
```

```
## [1] "2009-06-01"
```

timeSeries:

```
start(S)
```

```
## GMT
## [1] [2009-01-01]
```

```
end(S)
```

```
## GMT
## [1] [2009-06-01]
```

```
start(s)
```

```
## GMT
## [1] [2009-01-01]
```

```
end(s)
```

```
## GMT
## [1] [2009-06-01]
```

2.2.2 金融时间序列对象与 stats 包中的函数

所需 R 包

```
library(zoo)
library(xts)
library(timeSeries)
```

下面从 stats 包中选取一些与金融时间序列分析相关的函数进行介绍。

常规数据

测试数据为多元金融时间序列和单变量金融时间序列对象。

```
set.seed(1953)
data <- matrix(runif(18), ncol = 3)
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
colnames(Z) = paste("Z", 1:3, sep = ".")
Z
```

```
##                 Z.1       Z.2       Z.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
## 2009-04-01 0.4472982 0.9652389 0.6135854
## 2009-05-01 0.6602838 0.5931596 0.4981308
## 2009-06-01 0.2228321 0.2319316 0.2083319
```

xts:

```
X <- xts(data, as.Date(charvec))
colnames(X) = paste("X", 1:3, sep = ".")
X
```

```
##                 X.1       X.2       X.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
```

```
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
## 2009-04-01 0.4472982 0.9652389 0.6135854
## 2009-05-01 0.6602838 0.5931596 0.4981308
## 2009-06-01 0.2228321 0.2319316 0.2083319
```

timeSeries:

```
S <- timeSeries(data, charvec)
colnames(S) = paste("S", 1:3, sep = ".")
S
```

```
## GMT
##                   S.1       S.2       S.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
## 2009-04-01 0.4472982 0.9652389 0.6135854
## 2009-05-01 0.6602838 0.5931596 0.4981308
## 2009-06-01 0.2228321 0.2319316 0.2083319
```

zoo:

```
z <- Z[, 1]
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.5087459  0.4770248  0.1829114  0.4472982
## 2009-05-01 2009-06-01
##  0.6602838  0.2228321
```

xts:

```
x <- X[, 1]
x
```

```
##                  X.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
## 2009-04-01 0.4472982
## 2009-05-01 0.6602838
## 2009-06-01 0.2228321
```

timeSeries:

```
s <- S[, 1]
s
```

```
## GMT
##                  S.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
## 2009-04-01 0.4472982
## 2009-05-01 0.6602838
## 2009-06-01 0.2228321
```

2.2.2.1 能对金融时间序列对象应用 arima 函数么?

arima 函数的作用是对单变量时间序列对象建立 arima 模型。

zoo:

```
arima(z)
```

```
##
## Call:
```

```
## arima(x = z)
##
## Coefficients:
##       intercept
##          0.4165
## s.e.     0.0676
##
## sigma^2 estimated as 0.02744:  log likelihood = 2.27,  aic = -0.55
```

xts:

```
arima(x)
```

```
##
## Call:
## arima(x = x)
##
## Coefficients:
##       intercept
##          0.4165
## s.e.     0.0676
##
## sigma^2 estimated as 0.02744:  log likelihood = 2.27,  aic = -0.55
```

timeSeries:

```
arima(s)
```

```
##
## Call:
## arima(x = s)
##
## Coefficients:
##       intercept
```

```
##           0.4165
## s.e.      0.0676
##
## sigma^2 estimated as 0.02744:  log likelihood = 2.27,  aic = -0.55
```

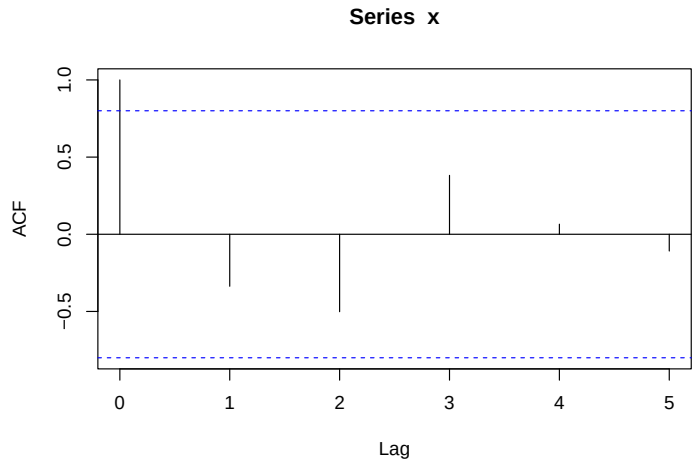
2.2.2.2 能对金融时间序列对象应用 acf 函数吗? acf 函数用以返回对象的自协方差和自相关序列。pacf 函数用以返回对象的偏自相关序列。ccf 函数用以返回互相关序列和互协方差序列。

zoo:

```
print(try(acf(z)))
```

xts:

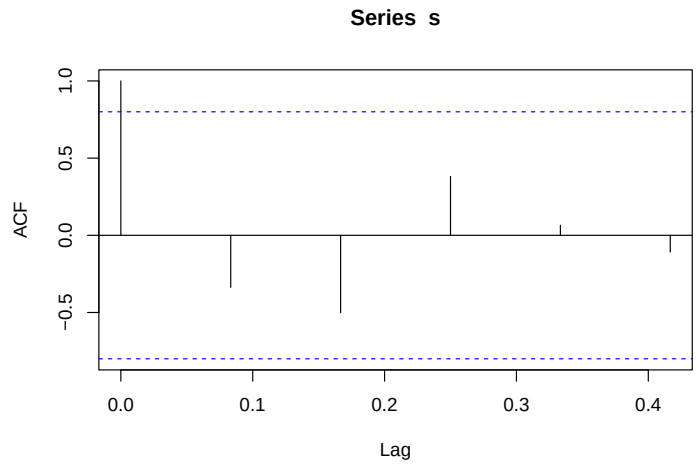
```
print(try(acf(x)))
```




```
##
## Autocorrelations of series 'x', by lag
##
##      0      1      2      3      4      5
##  1.000 -0.337 -0.502  0.382  0.065 -0.109
```

timeSeries:

```
print(acf(s))
```




```
##
## Autocorrelations of series 's', by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167
##  1.000 -0.337 -0.502  0.382  0.065 -0.109
```

2.2.2.3 能对金融时间序列对象应用 cov 函数吗? var 函数、cov 函数和 cor 函数用以返回对象 x 的方差、协方差或者 x，y 的协相关系数。

如果 x,y 是矩阵形式，其返回结果为对象 x 的列于对象 y 的列的协方差或者协相关系数。应用于 zoo、xts 和 timeSeries 对象是表现如下:

zoo:

```
var(z)
```

```
## [1] 0.0329245
```

```
var(Z)
```

```
##             Z.1        Z.2         Z.3
## Z.1 0.032924504 0.01578263 0.001619077
## Z.2 0.015782626 0.05390643 0.028639622
## Z.3 0.001619077 0.02863962 0.076510270
```

```
cov(Z)
```

```
##             Z.1        Z.2         Z.3
## Z.1 0.032924504 0.01578263 0.001619077
## Z.2 0.015782626 0.05390643 0.028639622
## Z.3 0.001619077 0.02863962 0.076510270
```

```
cor(Z)
```

```
##            Z.1       Z.2        Z.3
## Z.1 1.00000000 0.3746272 0.03225879
## Z.2 0.37462725 1.0000000 0.44595100
## Z.3 0.03225879 0.4459510 1.00000000
```

xts:

```
var(x)
```

```
##           X.1
## X.1 0.0329245
```

```
var(X)
```

```
##             X.1        X.2         X.3
## X.1 0.032924504 0.01578263 0.001619077
## X.2 0.015782626 0.05390643 0.028639622
## X.3 0.001619077 0.02863962 0.076510270
```

```
cov(X)
```

```
##             X.1        X.2         X.3
## X.1 0.032924504 0.01578263 0.001619077
## X.2 0.015782626 0.05390643 0.028639622
## X.3 0.001619077 0.02863962 0.076510270
```

```
cor(X)
```

```
##            X.1       X.2        X.3
## X.1 1.00000000 0.3746272 0.03225879
## X.2 0.37462725 1.0000000 0.44595100
## X.3 0.03225879 0.4459510 1.00000000
```

timeSeries:

```
var(s)
```

```
##           S.1
## S.1 0.0329245
```

```
var(S)
```

```
##             S.1        S.2         S.3
## S.1 0.032924504 0.01578263 0.001619077
## S.2 0.015782626 0.05390643 0.028639622
## S.3 0.001619077 0.02863962 0.076510270
```

```
cov(S)
```

```
##             S.1        S.2         S.3
## S.1 0.032924504 0.01578263 0.001619077
## S.2 0.015782626 0.05390643 0.028639622
## S.3 0.001619077 0.02863962 0.076510270
```

```
cor(S)
```

```
##            S.1       S.2        S.3
## S.1 1.00000000 0.3746272 0.03225879
## S.2 0.37462725 1.0000000 0.44595100
## S.3 0.03225879 0.4459510 1.00000000
```

2.2.2.4 能对金融时间序列对象应用 dist 函数吗? dist 函数用于返回对象的行对象的距离矩阵。

zoo:

```
dist(Z)
```

```
##            2009-01-01 2009-02-01 2009-03-01
## 2009-02-01  0.5329952
## 2009-03-01  0.3330673  0.5506632
## 2009-04-01  0.5075606  0.4475089  0.5220803
## 2009-05-01  0.4590844  0.2092585  0.6015870
## 2009-06-01  0.8491815  0.4666336  0.7489353
##            2009-04-01 2009-05-01
## 2009-02-01
## 2009-03-01
## 2009-04-01
## 2009-05-01  0.4439996
## 2009-06-01  0.8673840  0.6370503
```

```
dist(t(Z))
```

```
##           Z.1       Z.2
## Z.2 0.6732067
## Z.3 0.8383129 0.6047516
```

xts:

```
dist(X)
```

```
##            2009-01-01 2009-02-01 2009-03-01
## 2009-02-01  0.5329952
## 2009-03-01  0.3330673  0.5506632
## 2009-04-01  0.5075606  0.4475089  0.5220803
## 2009-05-01  0.4590844  0.2092585  0.6015870
## 2009-06-01  0.8491815  0.4666336  0.7489353
##            2009-04-01 2009-05-01
## 2009-02-01
## 2009-03-01
## 2009-04-01
## 2009-05-01  0.4439996
## 2009-06-01  0.8673840  0.6370503
```

```
dist(t(X))
```

```
##           X.1       X.2
## X.2 0.6732067
## X.3 0.8383129 0.6047516
```

timeSeries:

```
dist(S)
```

```
##            2009-01-01 2009-02-01 2009-03-01
## 2009-02-01  0.5329952
## 2009-03-01  0.3330673  0.5506632
## 2009-04-01  0.5075606  0.4475089  0.5220803
## 2009-05-01  0.4590844  0.2092585  0.6015870
## 2009-06-01  0.8491815  0.4666336  0.7489353
##            2009-04-01 2009-05-01
## 2009-02-01
```

```
## 2009-03-01
## 2009-04-01
## 2009-05-01  0.4439996
## 2009-06-01  0.8673840  0.6370503
```

```
dist(t(S))
```

```
##           S.1       S.2
## S.2 0.6732067
## S.3 0.8383129 0.6047516
```

2.2.2.5 能对金融时间序列对象应用 dnorm 对象吗? dnorm 函数用于返回对象所对应的正态分布的密度数据。相关参数分别由 mean 和 sd 指定。

zoo:

```
dnorm(z, mean = 0, sd = 1)
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.3505157  0.3560390  0.3923242  0.3609642
## 2009-05-01 2009-06-01
##  0.3208037  0.3891597
```

xts:

```
dnorm(x, mean = 0, sd = 1)
```

```
##                  X.1
## 2009-01-01 0.3505157
## 2009-02-01 0.3560390
## 2009-03-01 0.3923242
## 2009-04-01 0.3609642
## 2009-05-01 0.3208037
## 2009-06-01 0.3891597
```

timeSeries:

```
dnorm(s, mean = 0, sd = 1)
```

```
## GMT
##                   S.1
## 2009-01-01 0.3505157
## 2009-02-01 0.3560390
## 2009-03-01 0.3923242
## 2009-04-01 0.3609642
## 2009-05-01 0.3208037
## 2009-06-01 0.3891597
```

2.2.2.6 能对金融时间序列对象应用 filter 函数吗? filter 函数的作用是返回单变量时间序列对象的线性滤波序列，或者返回多变量时间序列对象的各变量的线性滤波序列。

zoo:

```
print(try(filter(z)))
```

xts:

```
print(try(filter(x)))
```

timeSeries:

```
filter(s, rep(1, 3))
```

```
## GMT
##                  S.1
## 2009-01-01        NA
## 2009-02-01 1.168682
## 2009-03-01 1.107234
```

```
## 2009-04-01 1.290493
## 2009-05-01 1.330414
## 2009-06-01       NA
```

2.2.2.7 能对金融时间序列对象应用 fivenum 函数吗? fivenum 用于返回 Tukey 五数数据（最小值、下 1/4 分位数、中位数、3/4 分位数、最大值）。

zoo:

```
fivenum(z)
```

```
## 2009-01-01 2009-02-01 2009-05-01 2009-06-01
##  0.5087459  0.4770248  0.6602838  0.2228321
```

xts:

```
fivenum(x)
```

```
##                  X.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-05-01 0.6602838
## 2009-06-01 0.2228321
```

timeSeries:

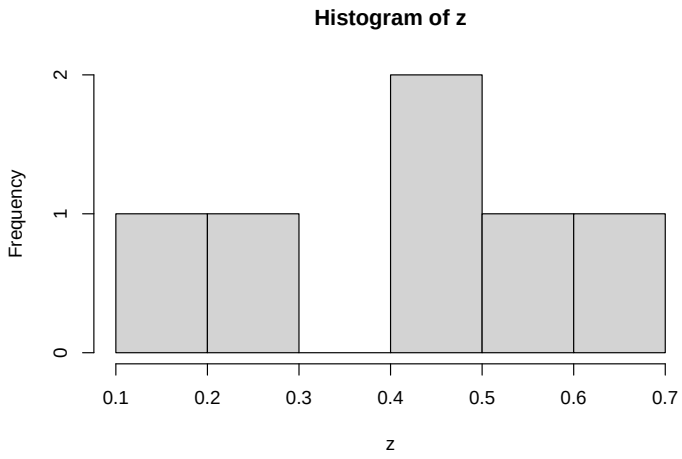
```
fivenum(s)
```

```
## [1] 0.5087459 0.4770248 0.3151048 0.6602838
## [5] 0.2228321
```

2.2.2.8 能对金融时间序列对象应用 hist 函数吗? hist 函数用以返回对象的直方图数据，如果参数 plot=TRUE，hist 函数返回的 histogram 对象将被 plot.histogram() 函数绘制为直方图。

zoo:

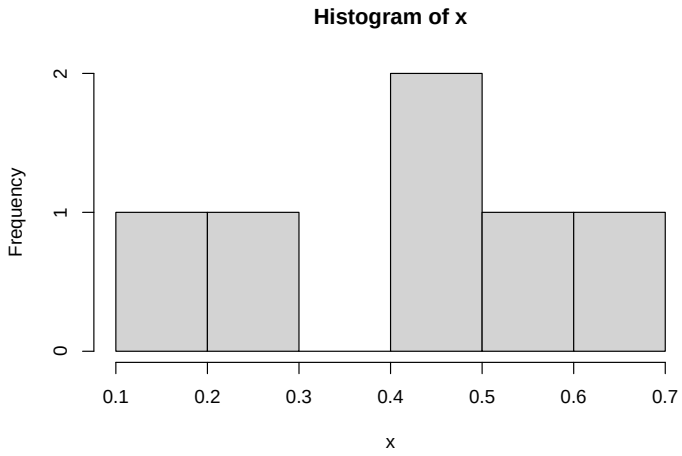
```
hist(z)$density
```




```
## [1] 1.666667 1.666667 0.000000 3.333333 1.666667
## [6] 1.666667
```

xts:

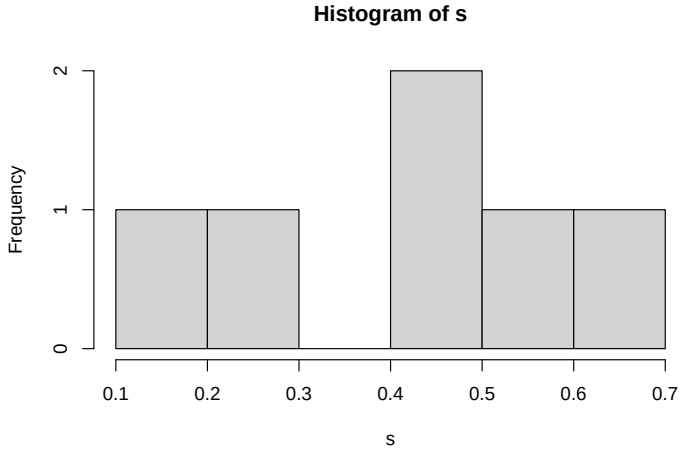
```
hist(x)$density
```




```
## [1] 1.666667 1.666667 0.000000 3.333333 1.666667
## [6] 1.666667
```

timeSeries:

```
hist(s)$density
```




```
## [1] 1.666667 1.666667 0.000000 3.333333 1.666667
## [6] 1.666667
```

2.2.2.9 能对金融时间序列对象应用 lag 函数吗? lag 函数用以返回时间序列的滞后序列。

zoo:

```
lag(Z)
```

```
##                  Z.1       Z.2       Z.3
## 2009-01-01 0.4770248 0.5736546 0.3990106
## 2009-02-01 0.1829114 0.5912718 0.8642168
## 2009-03-01 0.4472982 0.9652389 0.6135854
## 2009-04-01 0.6602838 0.5931596 0.4981308
## 2009-05-01 0.2228321 0.2319316 0.2083319
```

xts:

```
lag(X)
```

```
##                  X.1       X.2       X.3
## 2009-01-01        NA        NA        NA
## 2009-02-01 0.5087459 0.5740212 0.9310609
## 2009-03-01 0.4770248 0.5736546 0.3990106
## 2009-04-01 0.1829114 0.5912718 0.8642168
## 2009-05-01 0.4472982 0.9652389 0.6135854
## 2009-06-01 0.6602838 0.5931596 0.4981308
```

timeSeries:

```
lag(S)
```

```
## GMT
##               S.1[1]    S.2[1]    S.3[1]
## 2009-01-01        NA        NA        NA
## 2009-02-01 0.5087459 0.5740212 0.9310609
## 2009-03-01 0.4770248 0.5736546 0.3990106
```

```
## 2009-04-01 0.1829114 0.5912718 0.8642168
## 2009-05-01 0.4472982 0.9652389 0.6135854
## 2009-06-01 0.6602838 0.5931596 0.4981308
```

2.2.2.10 能对金融时间序列对象应用 lm 函数吗? lm 函数用以拟合线性模型。可以用来进行回归、单层次方差分析和协方差分析（而 aov 函数提供了一个更方面途径来做这件事）。

zoo:

```
fit <- lm(Z.1 ~ Z.2 + Z.3, data = Z)
fit
```

```
##
## Call:
## lm(formula = Z.1 ~ Z.2 + Z.3, data = Z)
##
## Coefficients:
## (Intercept)          Z.2          Z.3
##      0.2745       0.3514      -0.1104
```

```
resid(fit)
```

```
##  2009-01-01  2009-02-01  2009-03-01  2009-04-01
##  0.13533728  0.04501473 -0.20393808 -0.09863810
##  2009-05-01  2009-06-01
##  0.23236055 -0.11013639
```

xts:

```
fit <- lm(X.1 ~ X.2 + X.3, data = X)
fit
```

```
##
```

```
## Call:
## lm(formula = X.1 ~ X.2 + X.3, data = X)
##
## Coefficients:
## (Intercept)          X.2          X.3
##      0.2745       0.3514      -0.1104
```

```
resid(fit)
```

```
##  2009-01-01  2009-02-01  2009-03-01  2009-04-01
##  0.13533728  0.04501473 -0.20393808 -0.09863810
##  2009-05-01  2009-06-01
##  0.23236055 -0.11013639
```

timeSeries:

```
fit <- lm(S.1 ~ S.2 + S.3, data = S)
fit
```

```
##
## Call:
## lm(formula = S.1 ~ S.2 + S.3, data = S)
##
## Coefficients:
## (Intercept)          S.2          S.3
##      0.2745       0.3514      -0.1104
```

```
resid(fit)
```

```
##  2009-01-01  2009-02-01  2009-03-01  2009-04-01
##  0.13533728  0.04501473 -0.20393808 -0.09863810
##  2009-05-01  2009-06-01
##  0.23236055 -0.11013639
```

2.2.2.11 能对金融时间序列对象应用 lowess() 函数吗? lowess 函数的作用是基于局部加权多项式回归对对象进行 lowess 平滑。

zoo:

```
lowess(z)
```

```
## $x
## [1] 1 2 3 4 5 6
##
## $y
## [1] 0.5335548 0.4094979 0.3747239 0.4371960
## [5] 0.4124398 0.2685375
```

xts:

```
lowess(x)
```

```
## $x
## [1] 1 2 3 4 5 6
##
## $y
## [1] 0.5335548 0.4094979 0.3747239 0.4371960
## [5] 0.4124398 0.2685375
```

timeSeries:

```
lowess(s)
```

```
## $x
## [1] 1 2 3 4 5 6
##
## $y
## [1] 0.5335548 0.4094979 0.3747239 0.4371960
## [5] 0.4124398 0.2685375
```

2.2.2.12 能对金融时间序列对象应用 mad 函数吗? mad 函数用以计算中位数绝对偏差，即上中位数和下中位数从中位数的偏离程度。

zoo:

```
mad(z)
```

```
## [1] 0.181401
```

xts:

```
mad(x)
```

```
## [1] 0.181401
```

timeSeries:

```
mad(s)
```

```
## [1] 0.181401
```

2.2.2.13 能对金融时间序列对象应用 median 函数吗? median 函数的作用是返回简单中位数。

zoo:

```
median(x)
```

```
## [1] 0.4621615
```

xts:

```
median(x)
```

```
## [1] 0.4621615
```

timeSeries:

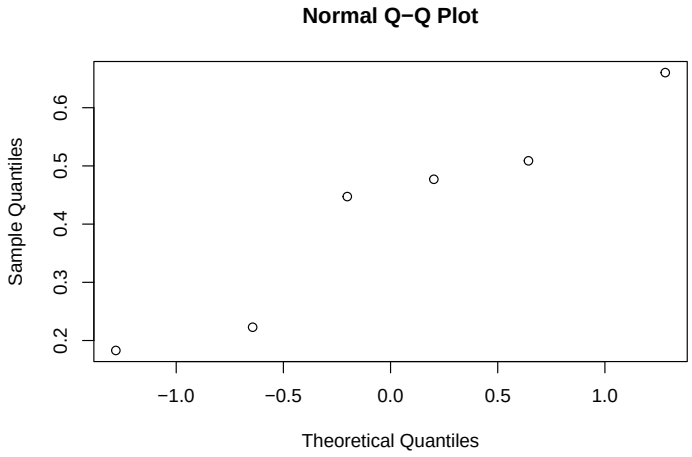
```
median(s)
```

```
## [1] 0.4621615
```

2.2.2.14 能对金融时间序列对象应用 qqnorm 函数吗? qqnorm 函数用以绘制对象的 QQ 图；qqline 函数用以添加 QQ 线。

zoo:

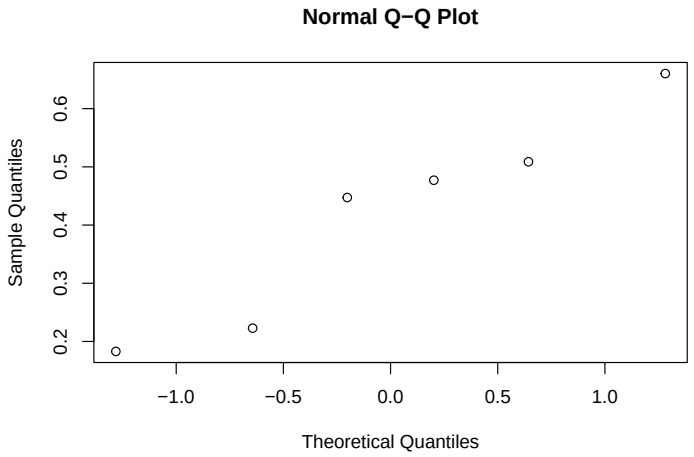
```
print(qqnorm(z))
```




```
## $x
## [1]  0.6433454  0.2018935 -1.2815516 -0.2018935
## [5]  1.2815516 -0.6433454
##
## $y
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.5087459  0.4770248  0.1829114  0.4472982
## 2009-05-01 2009-06-01
##  0.6602838  0.2228321
```

xts:

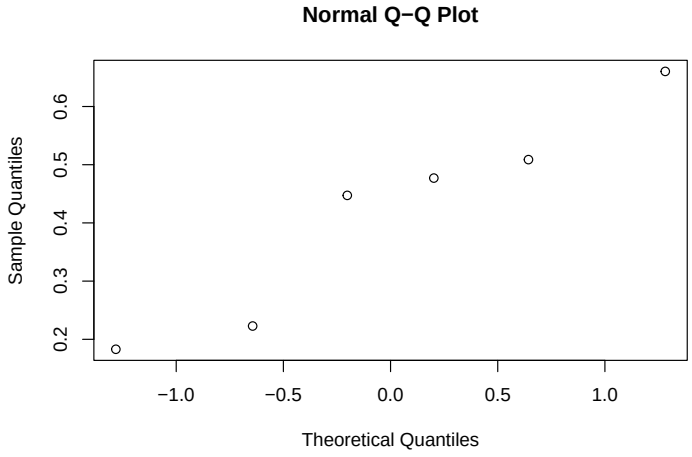
```
print(qqnorm(x))
```




```
## $x
## [1]  0.6433454  0.2018935 -1.2815516 -0.2018935
## [5]  1.2815516 -0.6433454
##
## $y
##                   X.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
## 2009-04-01 0.4472982
## 2009-05-01 0.6602838
## 2009-06-01 0.2228321
```

timeSeries:

```
print(qqnorm(s))
```




```
## $x
## [1]  0.6433454  0.2018935 -1.2815516 -0.2018935
## [5]  1.2815516 -0.6433454
##
## $y
## GMT
##                   S.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
## 2009-04-01 0.4472982
## 2009-05-01 0.6602838
## 2009-06-01 0.2228321
```

2.2.2.15 能对金融时间序列对象应用 smooth 函数吗? smooth 函数用以进行中位数平滑，该函数调用 Tukey 平滑器，如 3RS3R,3RSS,3R 等。

zoo:

```
smooth(z)
```

```
## 3RS3R Tukey smoother resulting from  smooth(x = z)
```

```
##  used 1 iterations
## [1] 0.5087459 0.4770248 0.4472982 0.4472982
## [5] 0.4472982 0.4472982
```

xts:

```
smooth(x)
```

```
## 3RS3R Tukey smoother resulting from  smooth(x = x)
##  used 1 iterations
## [1] 0.5087459 0.4770248 0.4472982 0.4472982
## [5] 0.4472982 0.4472982
```

timeSeries:

```
smooth(s)
```

```
## 3RS3R Tukey smoother resulting from  smooth(x = s)
##  used 1 iterations
## [1] 0.5087459 0.4770248 0.4472982 0.4472982
## [5] 0.4472982 0.4472982
```

2.2.2.16 能对金融时间序列对象应用 spectrum 函数吗?

spectrum 函数用以返回时间序列对象的谱密度。

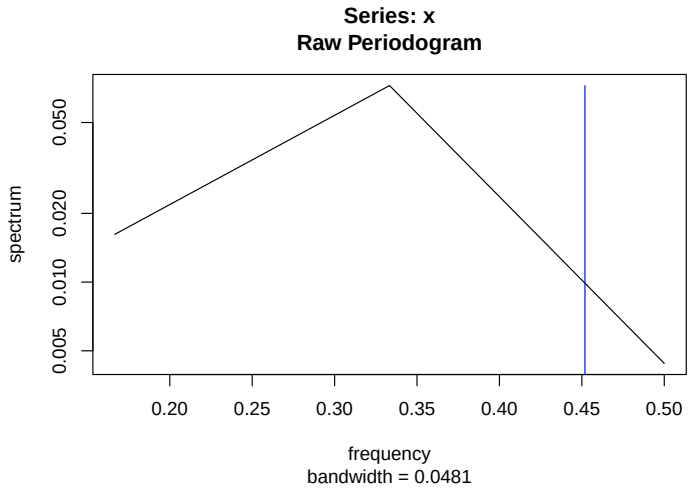
zoo:

```
print(try(spectrum(z)[1:2]))
```

```
## Error in na.fail.default(as.ts(x)) : 对象里有缺失值
## [1] "Error in na.fail.default(as.ts(x)) : 对象里有缺失值\n"
## attr(,"class")
## [1] "try-error"
## attr(,"condition")
## <simpleError in na.fail.default(as.ts(x)): 对象里有缺失值>
```

xts:

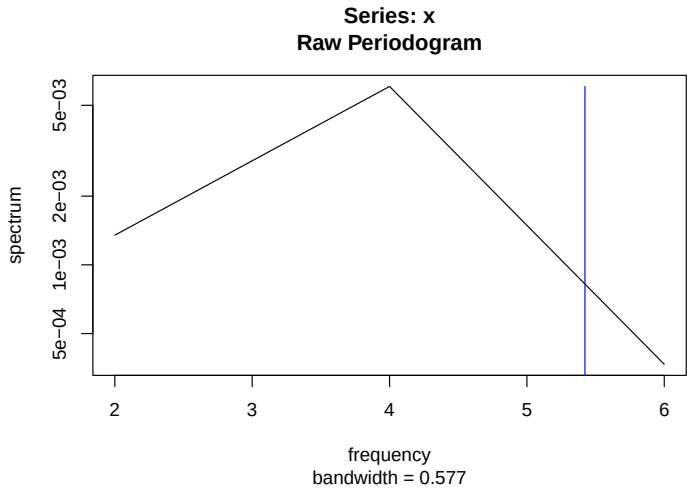
```
print(spectrum(x)[1:2])
```




```
## $freq
## [1] 0.1666667 0.3333333 0.5000000
##
## $spec
## [1] 0.016188013 0.072588922 0.004402901
```

timeSeries:

```
print(spectrum(s)[1:2])
```




```
## $freq
## [1] 2 4 6
##
## $spec
## [1] 0.0013490011 0.0060490769 0.0003669084
```

2.2.3 金融时间序列对象与 utils 包中的函数

所需 R 包。

```
library(zoo)
library(xts)
library(timeSeries)
```

选取 utils 包中与金融时间序列对象分析相关的函数进行介绍。

常规数据

```
set.seed(1953)
data <- matrix(runif(18), ncol = 3)
charvec <- paste("2009-0", 1:6, "-01", sep = "")
charvec
```

```
## [1] "2009-01-01" "2009-02-01" "2009-03-01"
## [4] "2009-04-01" "2009-05-01" "2009-06-01"
```

zoo:

```
Z <- zoo(data, as.Date(charvec))
colnames(Z) = paste("Z", 1:3, sep = ".")
Z
```

```
##                 Z.1       Z.2       Z.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
## 2009-04-01 0.4472982 0.9652389 0.6135854
## 2009-05-01 0.6602838 0.5931596 0.4981308
## 2009-06-01 0.2228321 0.2319316 0.2083319
```

xts:

```
X <- xts(data, as.Date(charvec))
colnames(X) = paste("X", 1:3, sep = ".")
X
```

```
##                 X.1       X.2       X.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
## 2009-04-01 0.4472982 0.9652389 0.6135854
## 2009-05-01 0.6602838 0.5931596 0.4981308
## 2009-06-01 0.2228321 0.2319316 0.2083319
```

timeSeries:

```
S <- timeSeries(data, charvec)
colnames(S) = paste("S", 1:3, sep = ".")
S
```

```
## GMT
##                  S.1       S.2       S.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
## 2009-04-01 0.4472982 0.9652389 0.6135854
## 2009-05-01 0.6602838 0.5931596 0.4981308
## 2009-06-01 0.2228321 0.2319316 0.2083319
```

zoo:

```
z <- Z[, 1]
z
```

```
## 2009-01-01 2009-02-01 2009-03-01 2009-04-01
##  0.5087459  0.4770248  0.1829114  0.4472982
## 2009-05-01 2009-06-01
##  0.6602838  0.2228321
```

xts:

```
x <- X[, 1]
x
```

```
##                  X.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
## 2009-04-01 0.4472982
## 2009-05-01 0.6602838
## 2009-06-01 0.2228321
```

timeSeries:

```
s <- S[, 1]
s
```

```
## GMT
##                   S.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
## 2009-04-01 0.4472982
## 2009-05-01 0.6602838
## 2009-06-01 0.2228321
```

2.2.3.1 能对金融时间序列对象应用 head 函数吗?

函数用于返回 vector、matrix、table、data frame 对象的头部部分和尾部部分。

ts:

```
ts <- rnorm(ts(rnorm(12)))
ts
```

```
##  [1]  0.4736224  0.4941460 -0.3687956  0.2365270
##  [5]  0.6201417  0.2098003 -1.4903779 -1.6032132
##  [9]  0.6542039 -0.1090904 -0.6485312  0.7964054
```

```
head(ts)
```

```
## [1]  0.4736224  0.4941460 -0.3687956  0.2365270
## [5]  0.6201417  0.2098003
```

对于规则的 ts 类型时间序列，属性信息会丢失。

zoo:

```
head(Z, 3)
```

```
##                  Z.1       Z.2       Z.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
```

```
head(z, 3)
```

```
## 2009-01-01 2009-02-01 2009-03-01
##  0.5087459  0.4770248  0.1829114
```

xts:

```
head(X, 3)
```

```
##                  X.1       X.2       X.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
```

```
head(x, 3)
```

```
##                  X.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
```

timeSeries:

```
head(S, 3)
```

```
## GMT
```

```
##                   S.1       S.2       S.3
## 2009-01-01 0.5087459 0.5740212 0.9310609
## 2009-02-01 0.4770248 0.5736546 0.3990106
## 2009-03-01 0.1829114 0.5912718 0.8642168
```

```
head(s, 3)
```

```
## GMT
##                   S.1
## 2009-01-01 0.5087459
## 2009-02-01 0.4770248
## 2009-03-01 0.1829114
```

2.3 性能测试

2.3.1 创建时间序列对象的性能测试

所需 R 包。

```
library(zoo)
library(xts)
library(timeSeries)
```

定义一个 systemTime 函数来度量各种操作的效率。

```
systemTime <- function(expr, gcFirst = TRUE, n = 20) {
    time <- sapply(integer(n), eval.parent(substitute(function(...) system.time(expr,
        gcFirst = gcFirst))))
    structure(apply(time, 1, median), class = "proc_time")
}
```

基于字符型时间戳，创建一个时期跨度为 100 年，列数为 5 的日时间序列对象需要多长时间?

R 会从 ASCII 格式，网页，或者 xls 与 csv 文件中读入时间序列数据后，通常将数据存储为字符串。

为了将字符串转化为 zoo、xts 或者 timeSeries 对象，需先将时间戳转化为合适的标签。考虑一个时期跨度为 100 年、共 35000 条记录的日时间序列数据。

常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
```

zoo:

```
systemTime(zoo(data, charvec))
```

```
##   用户   系统   流逝
## 0.066 0.001 0.071
```

xts:

```
print(try(xts(data, charvec)))
```

timeSeries:

```
systemTime(timeSeries(data, charvec))
```

```
##    用户    系统    流逝
## 0.2430 0.0040 0.2605
```

2.3.1.1 基于日期型时间戳，创建一个时期跨度为 100 年、列数为 5 的日时间序列对 象需要多长时间?

常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- as.Date(charvec)
class(index)
```

```
## [1] "Date"
```

zoo:

```
systemTime(zoo(data, index))
```

```
##    用户    系统    流逝
## 0.0070 0.0000 0.0075
```

这是基于 zoo 包创建时间序列对象的基本方法。

xts:

```
systemTime(xts(data, index))
```

```
##   用户   系统   流逝
## 0.001 0.000 0.001
```

这是基于 xts 包创建时间序列对象的基本做法。

timeSeries:

```
systemTime(timeSeries(data, index))
```

```
##   用户   系统   流逝
## 0.004 0.000 0.004
```

鉴于金融时间序列分析的对象大都是 timeDate 对象，下面的操作效率可以作为一个效率基准。

2.3.1.2 基于 GMT POSIXct 时间戳，创建一个时期跨度为 100 年、列数为 5 的日时间序列对象需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- as.POSIXct(charvec, tz = "GMT")
class(index)
```

```
## [1] "POSIXct" "POSIXt"
```

zoo:

```
systemTime(zoo(data, index))
```

```
##    用户    系统    流逝
## 0.0070 0.0000 0.0075
```

xts:

```
systemTime(xts(data, index))
```

```
##   用户   系统   流逝
## 0.001 0.000 0.001
```

timeSeries:

```
systemTime(timeSeries(data, index))
```

```
##   用户   系统   流逝
## 0.003 0.000 0.004
```

2.3.1.3 基于非 GMT POSIXct 时间戳，创建一个时期跨度为 100 年、列数为 5 的日时间序列对象需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
# index <-as.POSIXct(charvec, tz = 'CET') class(index)
```

zoo:

```
systemTime(zoo(data, index))
```

```
##   用户   系统   流逝
## 0.006 0.000 0.007
```

xts:

```
systemTime(xts(data, index))
```

```
##   用户   系统   流逝
## 0.001 0.000 0.001
```

timeSeries:

```
systemTime(timeSeries(data, index))
```

```
##   用户   系统   流逝
## 0.004 0.000 0.004
```

2.3.1.4 基于 timeDate 型时间戳，创建一个时期跨度为 100 年、列数为 5 的日时间序列对象需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- timeDate(charvec)
class(index)
```

```
## [1] "timeDate"
## attr(,"package")
## [1] "timeDate"
```

zoo:

```
systemTime(zoo(data, index))
```

```
##   用户   系统   流逝
## 0.007 0.000 0.008
```

xts:

```
systemTime(xts(data, index))
```

```
##   用户   系统   流逝
## 0.001 0.000 0.001
```

timeSeries:

```
systemTime(timeSeries(data, index))
```

```
##    用户    系统    流逝
## 0.0010 0.0000 0.0015
```

2.3.2 对时间序列取子集的操作性能

所需 R 包。

```
library(zoo)
library(xts)
library(timeSeries)
```

定义一个 systemTime 函数来度量各种操作性能。

```
systemTime <- function(expr, gcFirst = TRUE, n = 20) {
    time <- sapply(integer(n), eval.parent(substitute(function(...) system.time(expr,
        gcFirst = gcFirst))))
    structure(apply(time, 1, median), class = "proc_time")
}
```

2.3.2.1 基于整数，对时期跨度为 100 年，列数为 5 的日时间序列对象进行取子集操作需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- charvec
length <- floor(length(charvec)/2)
subset <- sample(charvec)[1:length]
```

zoo:

```
z <- zoo(data, as.Date(charvec))
systemTime(z[subset, ])
```

```
##  用户  系统  流逝
## 0.106 0.003 0.115
```

xts:

```
x <- xts(data, as.Date(charvec))
```

该操作耗时巨大!

timeSeries:

```
s <- timeSeries(data, charvec)
systemTime(s[subset, ])
```

```
##   用户   系统   流逝
## 0.136 0.003 0.145
```

2.3.2.2 基于 Date 对象，对时期跨度为 100 年，列数为 5 的日时间序列对象进行取子集操作需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- as.Date(charvec)
length <- floor(length(charvec)/2)
subset <- as.Date(sample(charvec)[1:length])
```

zoo:

```
z <- zoo(data, index)
systemTime(z[subset, ])
```

```
##    用户    系统    流逝
## 0.0060 0.0010 0.0075
```

xts:

```
x <- xts(data, index)
```

该操作耗时巨大。

timeSeries:

```
s <- timeSeries(data, index)
systemTime(s[subset, ])
```

```
##   用户   系统   流逝
## 0.007 0.000 0.008
```

2.3.2.3 基于 Date 对象，对一个时期跨度为 100 年，列数为 5 的 GMT POSIXct 格式的日时间序列进行取子集操作需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- as.POSIXct(charvec, tz = "GMT")
length <- floor(length(charvec)/2)
subset <- as.POSIXct(sample(charvec)[1:length], tz = "GMT")
```

zoo:

```
z <- zoo(data, index)
systemTime(z[subset, ])
```

```
##    用户    系统    流逝
## 0.0070 0.0010 0.0095
```

xts:

```
x <- xts(data, index)
```

该操作耗时巨大!

timeSeries:

```
s <- timeSeries(data, index)
systemTime(s[subset, ])
```

```
##  用户  系统  流逝
## 0.007 0.000 0.008
```

2.3.2.4 基于 Date 对象，对一个时期跨度为 100 年，列数为 5 的非 GMT POSIXct 格式的日时间序列进行取子集操作需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- as.POSIXct(charvec, format = "%Y-%m-%d", tz = "CET")

length <- floor(length(charvec)/2)
subset <- as.POSIXct(sample(charvec)[1:length], tz = "GMT")
```

zoo:

```
z <- zoo(data, index)
```

```
## Warning in zoo(data, index): some methods for
## "zoo" objects do not work if the index entries in
## 'order.by' are not unique
```

```
systemTime(z[subset, ])
```

```
##  用户  系统  流逝
## 3e-03 5e-04 3e-03
```

xts:

```
x <- xts(data, order.by = index)
```

该操作耗时巨大!

timeSeries:

```
s <- timeSeries(data, index)
```

这里有问题。

2.3.2.5 基于 Date 对象，对一个时期跨度为 100 年，列数为 5 的以 GMT timeDate 对象形式存储的日时间序列进行取子集操作需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- timeDate(charvec)
length <- floor(length(charvec)/2)
subset <- timeDate(sample(charvec)[1:length])
```

zoo:

```
z <- zoo(data, index)
systemTime(z[subset, ])
```

```
##   用户   系统   流逝
## 0.007 0.001 0.009
```

zoo 会自动经 fCalendar 包调用旧版本的 timeDate 函数，因此要 detach。

xts:

该操作耗时巨大!

timeSeries:

```
s <- timeSeries(data, index)
systemTime(s[subset, ])
```

```
##    用户    系统    流逝
## 0.0055 0.0000 0.0070
```

2.3.2.6 基于 Date 对象，对一个时期跨度为 100 年，列数为 5 的以 GMT timeDate 对象形式存储的日时间序列进行取子集操作需要多长时间? 常规数据

```
charvec <- format(seq(from = as.Date("1901-01-01"), to = as.Date("1999-12-31"),
    by = "day"))
data <- matrix(rnorm(length(charvec) * 5), ncol = 5)
index <- timeDate(charvec, zone = "Zurich", FinCenter = "Zurich")
length <- floor(length(charvec)/2)
subset <- timeDate(sample(charvec)[1:length], zone = "Zurich",
    FinCenter = "Zurich")
```

zoo:

```
z <- zoo(data, index)
systemTime(z[subset, ])
```

```
##   用户   系统   流逝
## 0.007 0.001 0.009
```

xts:

```
x <- xts(data, index)
# systemTime(x[subset, ])
```

timeSeries:

```
s <- timeSeries(data, index)
systemTime(s[subset, ])
```

```
##   用户   系统   流逝
## 0.006 0.000 0.007
```

3 应用篇

3.1 ARIMA 模型

3.2 GARCH 模型

3.3 VaR 模型

3.4 极值理论模型