NOTE on TimeSeries & R

zy

2020 年 10 月 28 日

目录

Chapter 1

时间序列概念

1.1 时间序列的分解

1.1.1 例: 北京地区洪涝灾害数据

```
> flood.data <- matrix(c(
+   1949 , 1 , 331.12 , 243.96 ,
+   1950 , 2 , 380.44 , 293.90 ,
+   1951 , 3 , 59.63 , 59.63,
+   1952 , 4 , 37.89 , 18.09,
+   1953 , 5 , 103.66 ,72.92,
+   1954 , 6 , 316.67 , 244.57,
+   1955 , 7 , 208.72 , 155.77,
+   1956 , 8 , 288.79 , 255.22,
+   1957 , 9 , 25.00 , 0.50,
+   1958 , 10 , 84.72 , 48.59,
+   1959 , 11 , 260.89 ,202.96,
+   1960, 12 , 27.18 ,15.02,
+   1961, 13 , 20.74 ,17.09,
+   1962 , 14 , 52.99 ,14.66,
+   1963 , 15 , 99.25 , 45.61,
+   1964 , 16 , 55.36 ,41.90),
+   byrow=T, ncol=4,
+   dimnames=list(1949:1964, c("year", "t", "area1", "area2")))
```

matrix(data, nrow, ncol, byrow, dimnames)

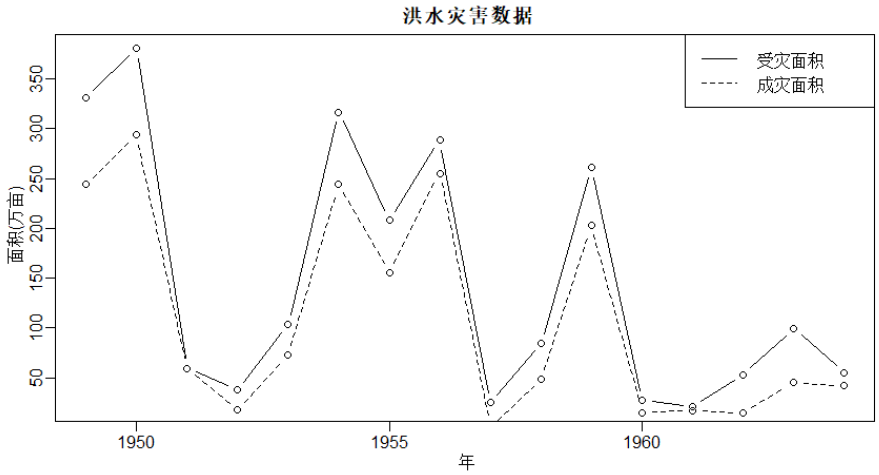
- 数据是成为矩阵的数据元素的输入向量。
- nrow 是要创建的行数。ncol 是要创建的列数。
- byrow 是一个逻辑线索。如果为 TRUE，则输入向量元素按行排列。
- dimname 是分配给行和列的名称。

```
> flood.area1 <- ts(flood.data[,"area1"],
+                   frequency=1,
+                   start=1949)
> flood.area2 <- ts(flood.data[,"area2"],
+                   frequency=1,
+                   start=1949)
```

timeseries.object.name <- ts(data, start, end, frequency)

- data 是包含在时间序列中使用的值的向量/矩阵。除了参数 data，所有其他参数是可选的。
- start 以时间序列指定第一次观察的开始时间。
- end 指定时间序列中最后一次观测的结束时间。
- frequency 指定每单位时间的观测数。

```
> fig.flood <- function(){
+   opar <- par(mar=c(3,3,2,1), mgp=c(1.5,0.5,0))
+   on.exit(par(opar))
+   plot(flood.area1, lty=1, type="b",
+        main='洪水灾害数据',
+        xlab='年', ylab='面积(万亩)')
+   lines(flood.area2, lty=2, type="b")
+   legend("topright", lty=c(1,2), legend=c('受灾面积', '成灾面积'))
+ }
> fig.flood()
```



在 R 语言做图中，可以简单的通过配置参数达到想要的效果，但是参数有很多，有必要进行分类，避免滥用或浪费。比如有一些参数如颜色大小是可以通用的，被分到了 par 里面，作为公共参数集合；还有一些如坐标轴就只能有类似 plot 这样的函数保有，给别人人家也用不到那些啊。如果 plot 的坐标轴要用颜色相关的属性，那么就可以直接去 par 中取来用就是了。如果 title 想用字体这个属性，也可以去 par 中取。所以 par 理所当然的可以被称为公共参数列表了。

opar <- par(no.readonly=TRUE) 这个语句。这个语句保存了之前的默认参数.

opar <- par(mar=c(3,3,2,1), mgp=c(1.5,0.5,0))

mar

A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 4, 4, 2) + 0.1. 当出现轴标题与轴标注发生重叠时，可以手动设置 mar 来解决.

使用 **mgp**=c(5,1,0) 把轴标题设置距离图形区域 5，轴标签设置为 1，轴本身为 0，单位为行高【默认 mgp=c(3, 1, 0)】

on.exit(expr = NULL, add = FALSE, after = TRUE)

- expr: an expression to be executed.
- add: if TRUE, add expr to be executed after any previously set expressions (or before if after is FALSE); otherwise (the default) expr will overwrite any previously set expressions.
- after: if add is TRUE and after is FALSE, then expr will be added on top of the expressions that were already registered. The resulting last in first out order is useful for freeing or closing resources in reverse order.

plot(v,type,col,xlab,ylab)

- v 是包含數值的向量。
- 類型採用值“p”僅替換點，“l”僅替換線和“o”替換點和線。
- xlab 是 x 軸的標籤。ylab 是 y 軸的標籤。
- main 是圖表的標題。
- col 用作給點和線的顏色。

通過使用 **lines()** 函數，可以在同一個圖表上繪製多條線。

legend(x, y = NULL, legend, fill = NULL, col = par("col"), border = "black", lty, lwd, pch, angle = 45, density = NULL, bty = "o", bg = par("bg"), box.lwd = par("lwd"), box.lty = par("lty"), box.col = par("fg"))

Arguments | 参数

x, y: 用于定位图例，也可用单键词"bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center"

legend: 字符或表达式向量

fill: 用特定的颜色进行填充

col: 图例中出现的点或线的颜色

border: 当 fill = 参数存在的情况下，填充色的边框

lty, lwd: 图例中线的类型与宽度

pch: 点的类型

angle: 阴影的角度

density: 阴影线的密度

bty: 图例框是否画出，o 为画出，默认为 n 不画出

bg:bty != "n" 时，图例的背景色

box.lty, box.lwd, box.col

bty = "o" 时，图例框的类型，box.lty 决定是否为虚线，box.lwd 决定粗线，box.col : 决定颜色

Example | 例子

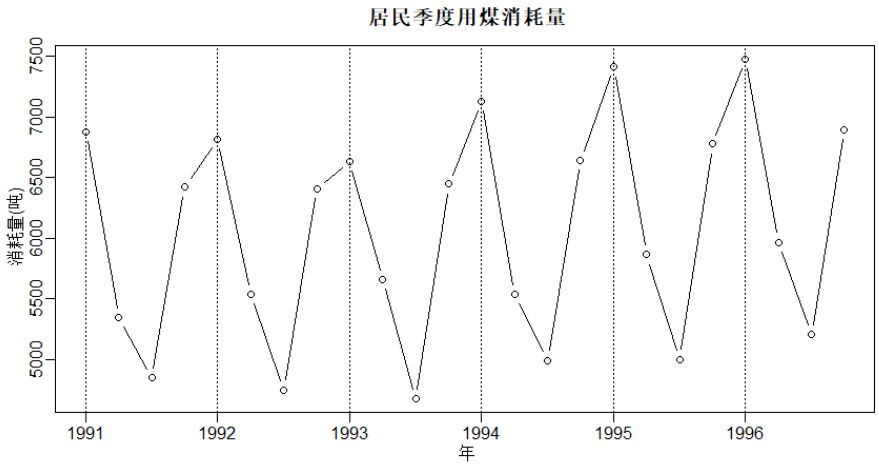
> legend("topleft", inset=.05, title="Drug Type", c("A","B"), + lty=c(1, 2), pch=c(15, 17), col=c("red", "blue"))

1.1.2 例: 居民用煤消耗季度值

```
> coal.consumption <-
+     ts(c(
+         6878.4 , 5343.7 , 4847.9 , 6421.9 ,
+         6815.4 , 5532.6 , 4745.6 , 6406.2 ,
```

```
+          6634.4 , 5658.5 , 4674.8 , 6445.5 ,
+          7130.2 , 5532.6 , 4989.6 , 6642.3 ,
+          7413.5 , 5863.1 , 4997.4 , 6776.1 ,
+          7476.5 , 5965.5 , 5202.1 , 6894.1 ),frequency=4, start=c(1991,1))
```

```
> demo.coal.data <- function(){
+      opar <- par(mar=c(3,3,3,1), mgp=c(1.5,0.5,0))
+      on.exit(par(opar))
+      plot(coal.consumption, lty=1, type='b',
+           main=' 居民季度用煤消耗量',
+           xlab=' 年', ylab=' 消耗量(吨)')
+      abline(v=1991:1996, lty=3)
+ }
> demo.coal.data()
```



1.1.2.1 用每年的平均值作为趋势项估计, 季度平均作为季节项

这样的趋势是阶梯函数，每年的 4 个季度的趋势相等。去趋势后同季度平均作为季节项。

```
> y <- coal.consumption
> ymore <- ts(c(y, rep(NA,4)), start=start(y), frequency=4)
> ymat <- matrix(c(y), byrow=TRUE, nrow=6, ncol=4)
> cols <- rainbow(20)
> ic <- 1


> ## 用同季度的值平均得到四个季节项--季节项得求取函数设计
> get.season <- function(yd){ # input: Detrended series
+      ymat <- matrix(c(yd), byrow=TRUE, ncol=4)
+      ## 季节项求取
+      seas0 <- apply(ymat, 2, mean, na.rm=TRUE)
```

```
        +     seas0
        + }

> ## 画去除趋势后的序列、季节项和随机项
> plot.season <- function(yd, seas0){ # input Detrended series用已除趋势项的序列
        +     ## 季节项
        +     seas <- rep(seas0, 6)
        +     seas <- ts(seas, start=c(1991,1), frequency=4)
        +     ## 随机项
        +     r <- c(yd) - seas
        +     r <- ts(r, frequency=4, start=c(1991,1))
        +     plot(yd, type='b', lwd=2,
        +          main="Detrended Series(black), Seasonal(red) and Random(cyan)",
        +          xlim=c(1991,1998), ylab="")
        +     abline(v=1991:1996, col="gray")
        +     abline(h=0, col="gray")
        +     lines(seas, type="l", col="red")
        +     lines(r, type="l", col="cyan")
        + }
```

rep(x, time = , length = , each = ,)

- x：代表的是你要进行复制的对象，可以是一个向量或者是一个因子。
- times：代表的是复制的次数，只能为正数。负数以及 NA 值都会为错误值。复制是指的是对整个向量进行复制。
- each：代表的是对向量中的每个元素进行复制的次数。
- length.out：代表的是最终输出向量的长度。

rainbow(n, s = 1, v = 1, start = 0, end = max(1, n - 1)/n, alpha = 1)
例如，基于彩虹色渐变获取 10 个颜色, 依次由 red、yellow、green、cyan、green、blue 、magenta 等顺序细化而成：rainbow(10)

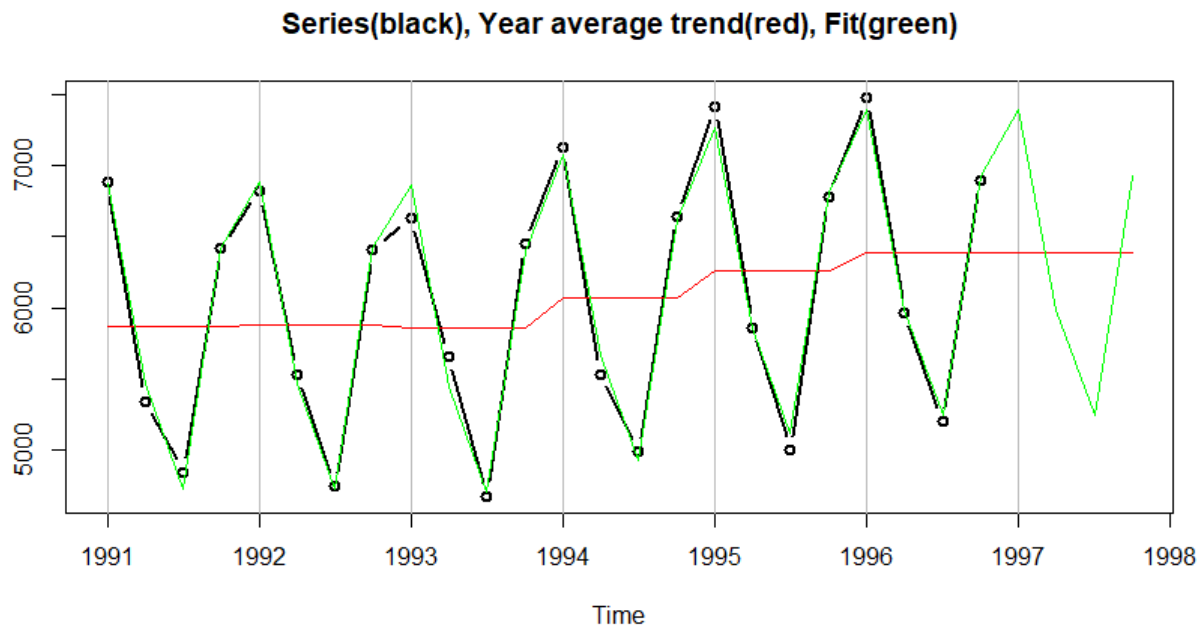
apply(array, margin, FUN, ...)

- 在 array 上，沿 margin 方向，依次调用 FUN。返回值为 vector。
- margin 表示数组引用的第几维下标（即 array[index1, index2, ...] 中的第几个 index），1 对应为 1 表示行，2 表示列，c(1,2) 表示行列。

下图为原始序列、趋势、拟合（包括趋势与季节项）：

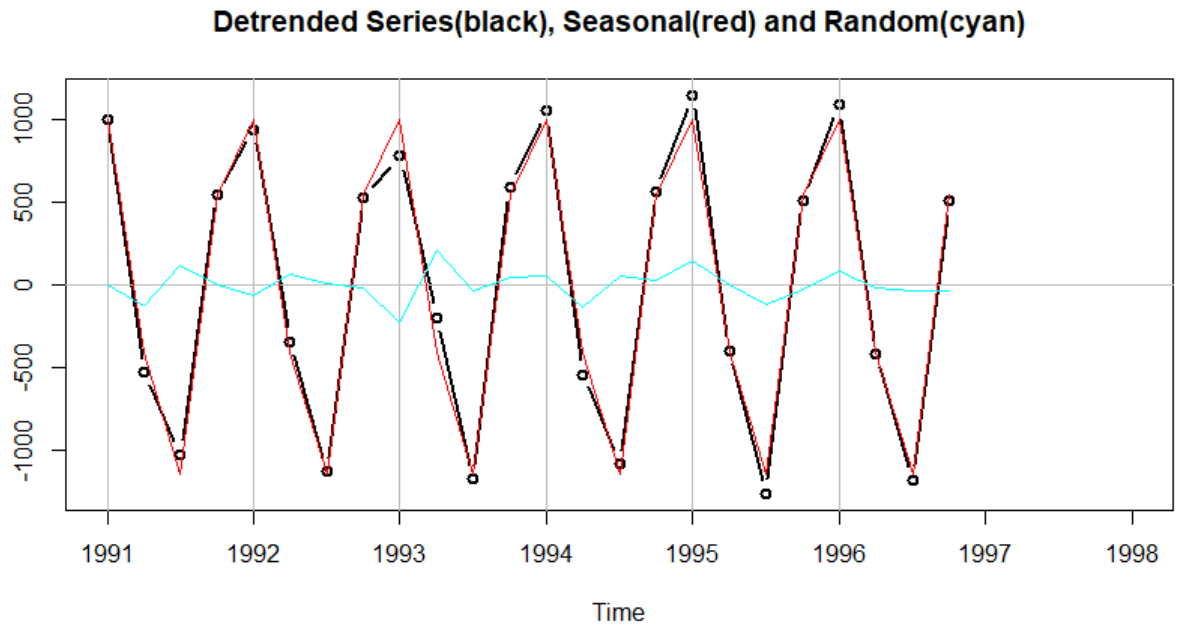
```
> tr0 <- apply(ymat, 1, mean, na.rm=TRUE)          #趋势项
```

```
> tr <- rep(tr0, each=4)
> tr <- ts(tr, frequency=4, start=c(1991,1))

> y.detrended <- y - tr                                  #去除趋势
> seas0 <- get.season(y.detrended)                       #求季节项

> tr.more <- ts(c(tr, rep(tr[length(tr)],4)),
+               start=start(y), frequency=4)
> seas.more <- ts(rep(seas0, 7),
+                 start=start(y), frequency=4)
> y.pred <- tr.more + seas.more
> plot(ymore,
+      main="Series(black), Year average trend(red), Fit(green)",
+      lwd=2,
+      type="b", col="black",
+      ylab="")
> abline(v=1991:1997, col="gray")
> lines(tr.more, col="red", type="l")
> lines(y.pred, col="green", type="l")
```



下图为去掉了趋势后的序列、季节项、取掉了趋势与季节项后的随机项：

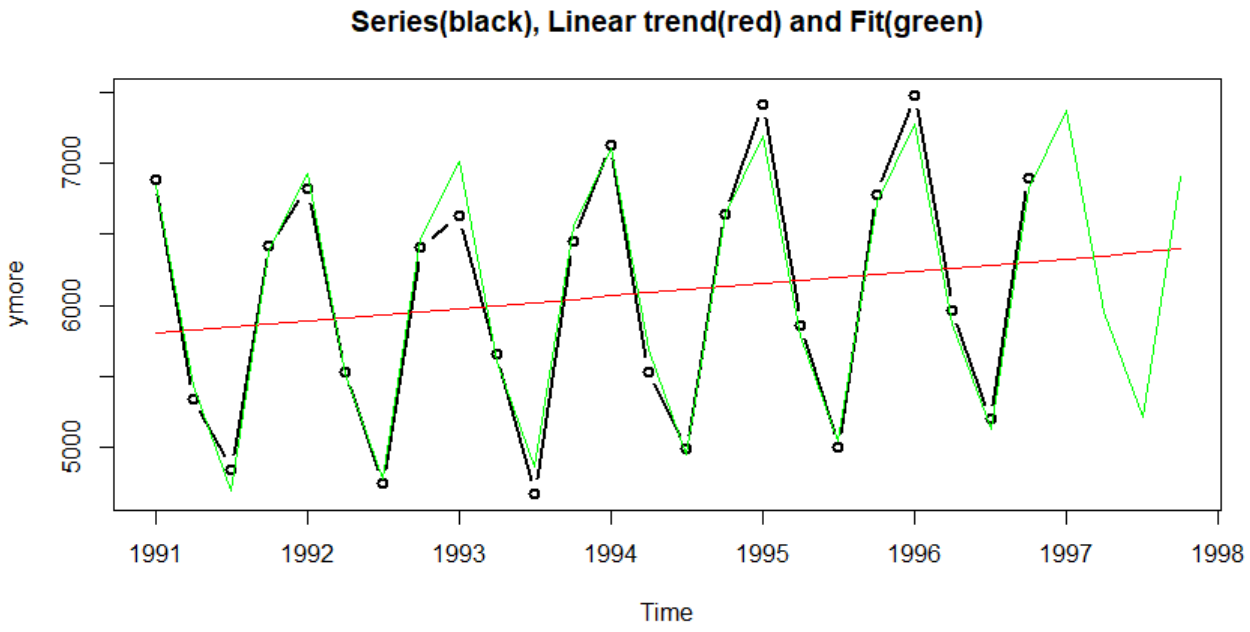
```
> plot.season(y.detrended, seas0)
```



1.1.2.2 用时间的线性回归作为趋势项估计，季度平均作为季节项

可以设趋势为 $T_t = a + bt$，用一元线性回归程序估计趋势并预测。下图绘制了原始数据、估计的趋势、拟合值（包括趋势与季节项）：

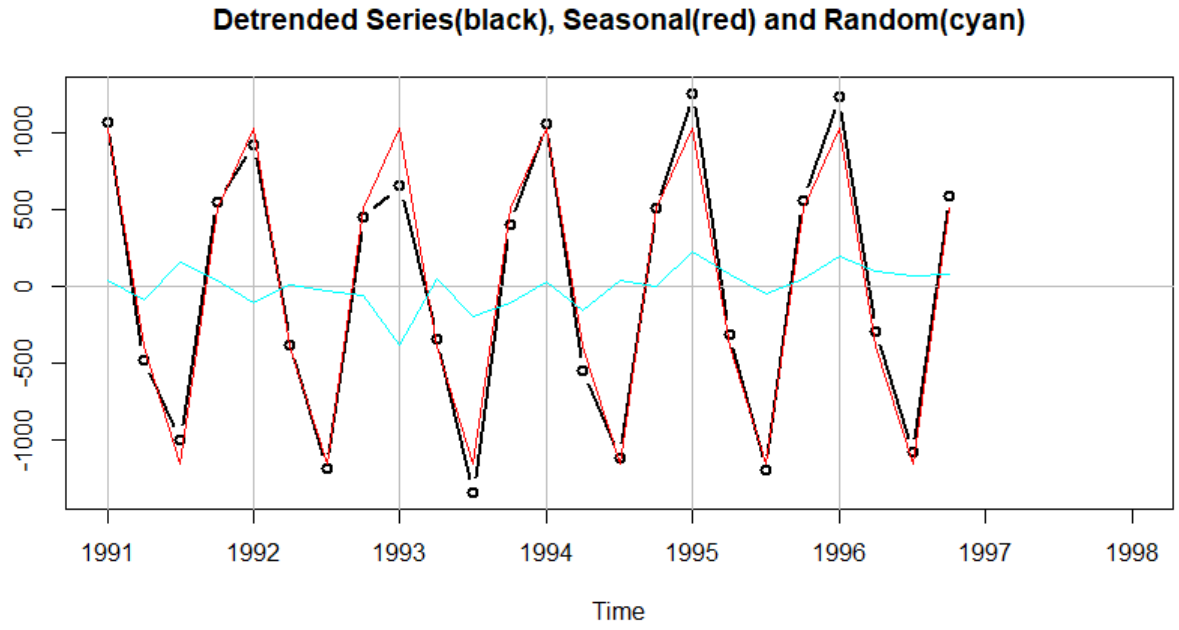
```
> yy <- c(y)
> tt <- seq(length(y))
> lmr <- lm(yy ~ tt)
> tr.more <- ts(predict(lmr, newdata=list(tt=seq(length(y)+4))),
+               frequency=4, start=c(1991,1))
> ## season
> y.detrended <- y - tr.more[1:length(y)]
> seas0 <- get.season(y.detrended)
> seas.more <- ts(rep(seas0, 7),
+                 start=start(y), frequency=4)
> y.pred <- tr.more + seas.more
> plot(ymore, main="Series(black), Linear trend(red) and Fit(green)",
+      lwd=2,
+      type="b", col="black")
> lines(tr.more, col="red", type="l")
> lines(y.pred, col="green", type="l")
```



lm() 是 R 语言中经常用到的函数，用来拟合回归模型。它是拟合线性模型最基本的函数。

下图为去掉了趋势后的序列、季节项、取掉了趋势与季节项后的随机项：

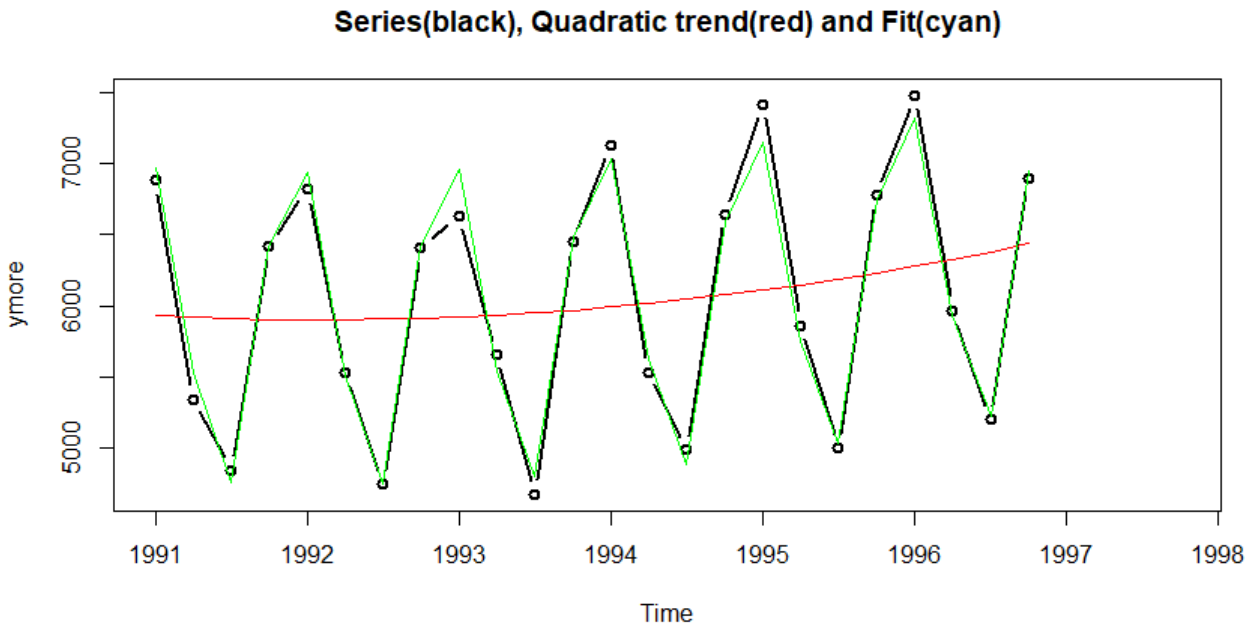
```
> plot.season(y.detrended, seas0)
```



1.1.2.3 二次曲线趋势

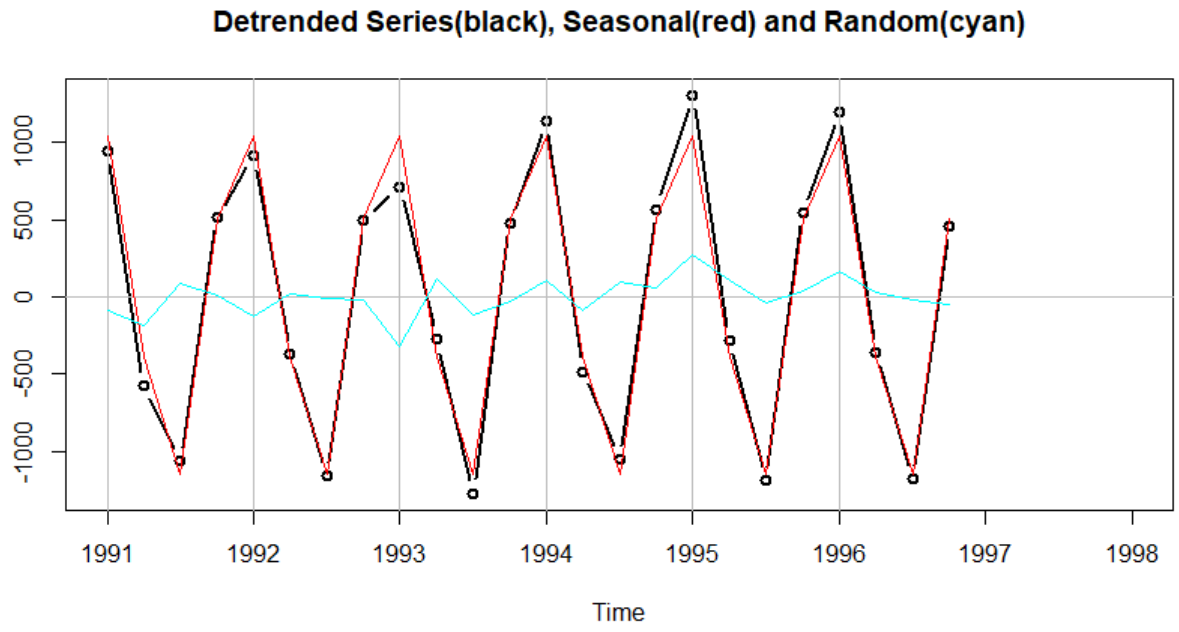
可以用 $T_t = a + bt + ct^2$ 作为趋势模型，用多元线性回归程序估计模型并预测。下图绘制了原始数据、估计的趋势、拟合值（包括趋势与季节项）：

```
> yy <- c(y)
> tt <- seq(length(y))
> lmr <- lm(yy ~ tt + I(tt^2))
> tr.more <- ts(predict(lmr, new.data=list(tt=seq(length(y)+4))),
+               frequency=4, start=c(1991,1))
> ## season
> y.detrended <- y - tr.more[1:length(y)]
> seas0 <- get.season(y.detrended)
> seas.more <- ts(rep(seas0, 7),
+                 start=start(y), frequency=4)
> y.pred <- tr.more + seas.more
> plot(ymore, main="Series(black), Quadratic trend(red) and Fit(cyan)",
+      lwd=2,
+      type="b", col="black")
> lines(tr.more, col="red", type="l")
> lines(y.pred, col="green", type="l")
```



下图为去掉了趋势后的序列、季节项、取掉了趋势与季节项后的随机项：

```
> plot.season(y.detrended, seas0)
```

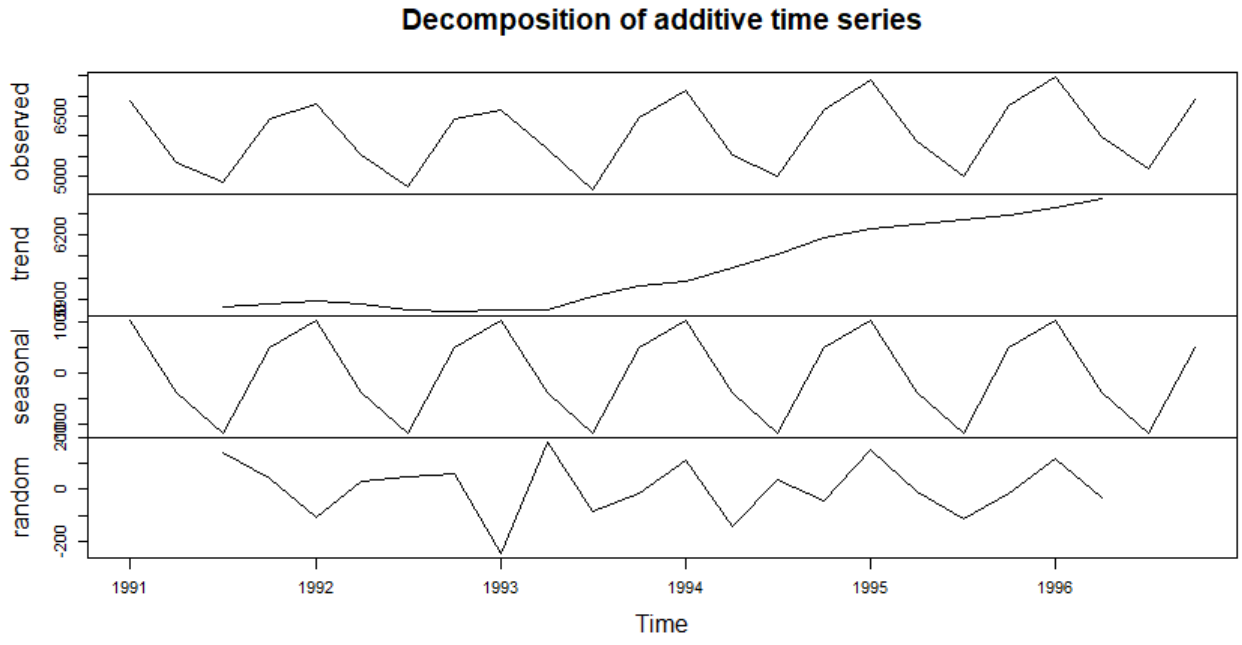


1.1.2.4 decompose() 函数

R 的 stats 包的 **decompose()** 函数输入一个时间序列，将其分解为趋势项、季节项和随机项。去趋势的方法是中心对称滑动平均。可以用 type="additive" 或 type="multiplicative" 指定各项之间是相加还是相乘。

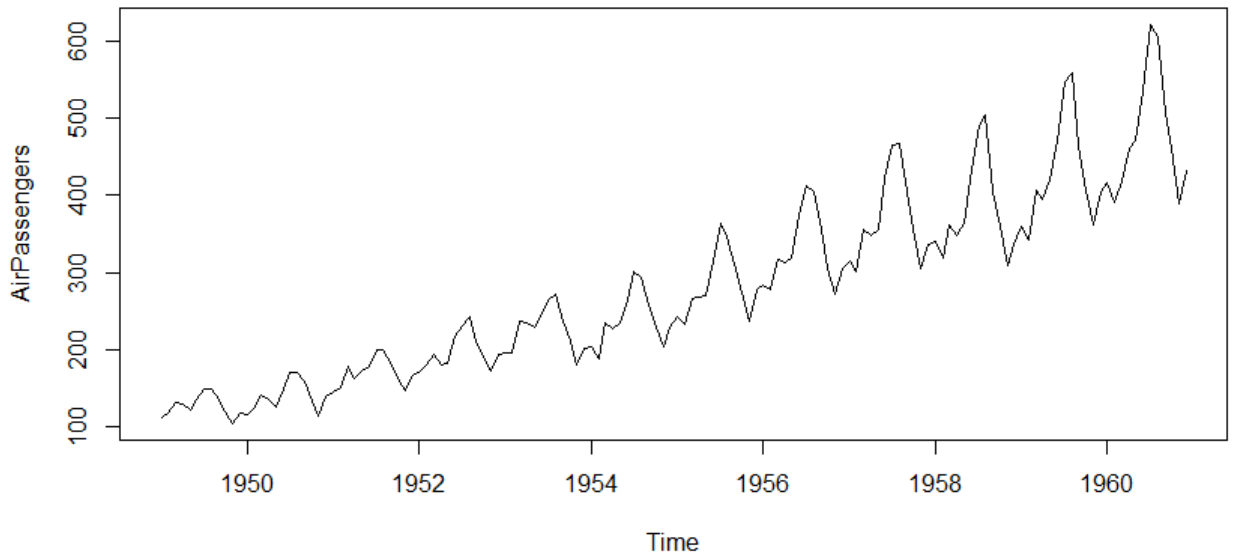
下图分别是原始序列、趋势估计、季节项估计和随机项估计。函数输出结果为一个列表，各项为 x, seasonal, trend, random 等。

```
> res1 <- decompose(coal.consumption)
> plot(res1)
```



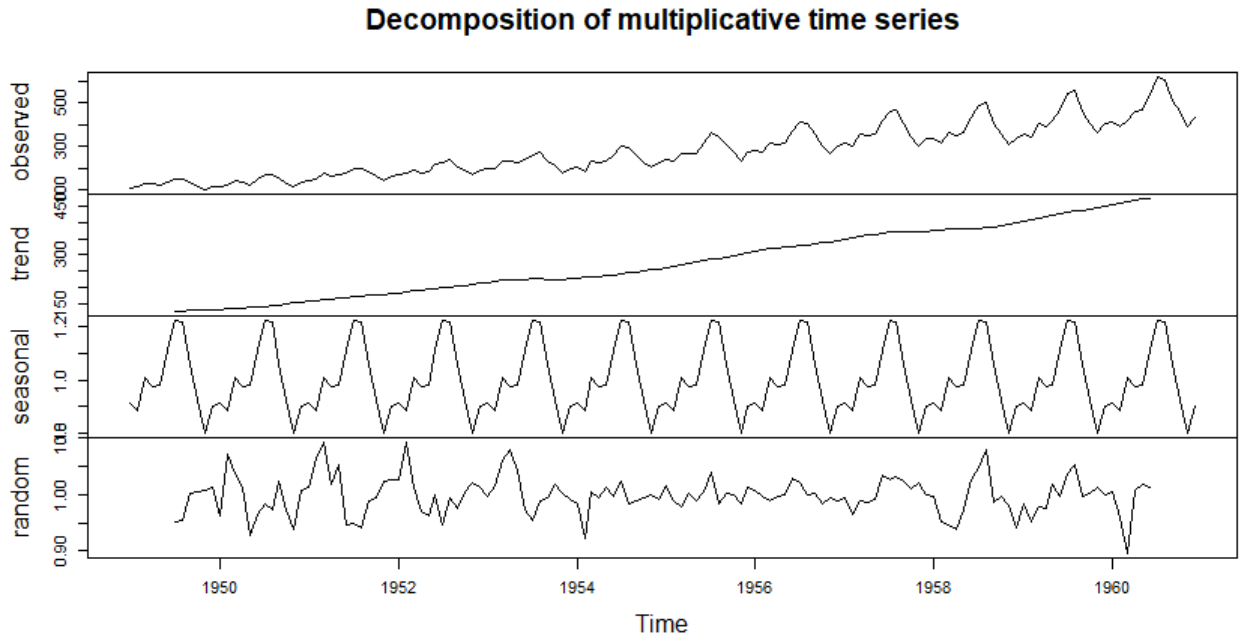
又比如，考虑著名的美国泛美航空公司 1949-1960 的国际航班订票数的月度数据（单位：千人），12 年 144 个月的数据。序列图：

```
> plot(AirPassengers)
```



分解:

```
> plot(decompose(AirPassengers, type="multiplicative"))
```

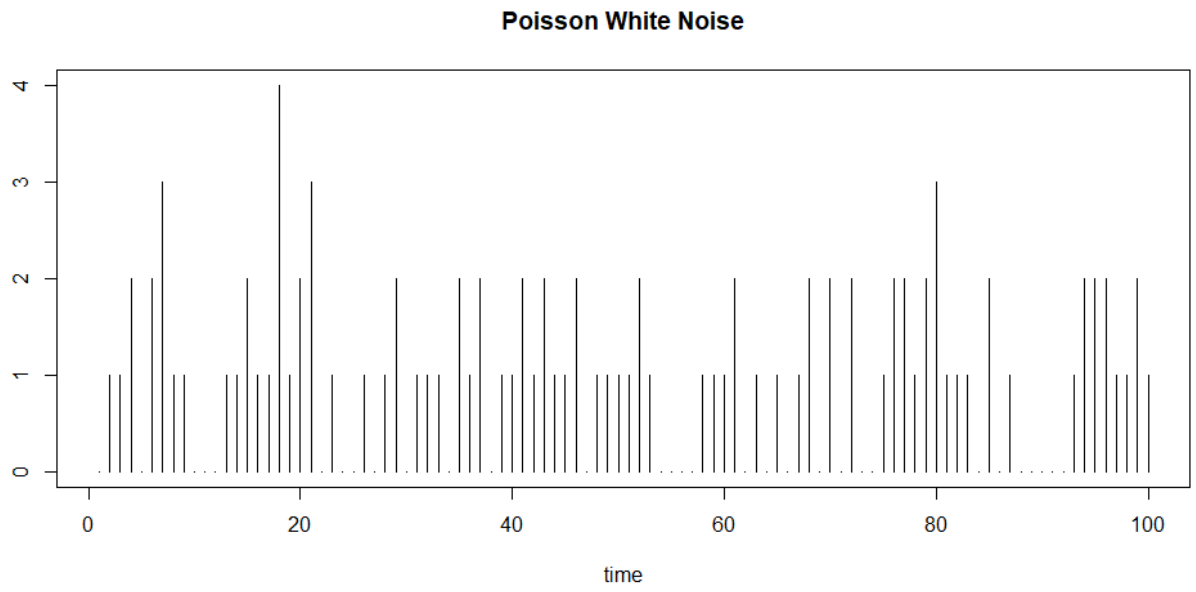


1.2 平稳序列

1.2.1 白噪声

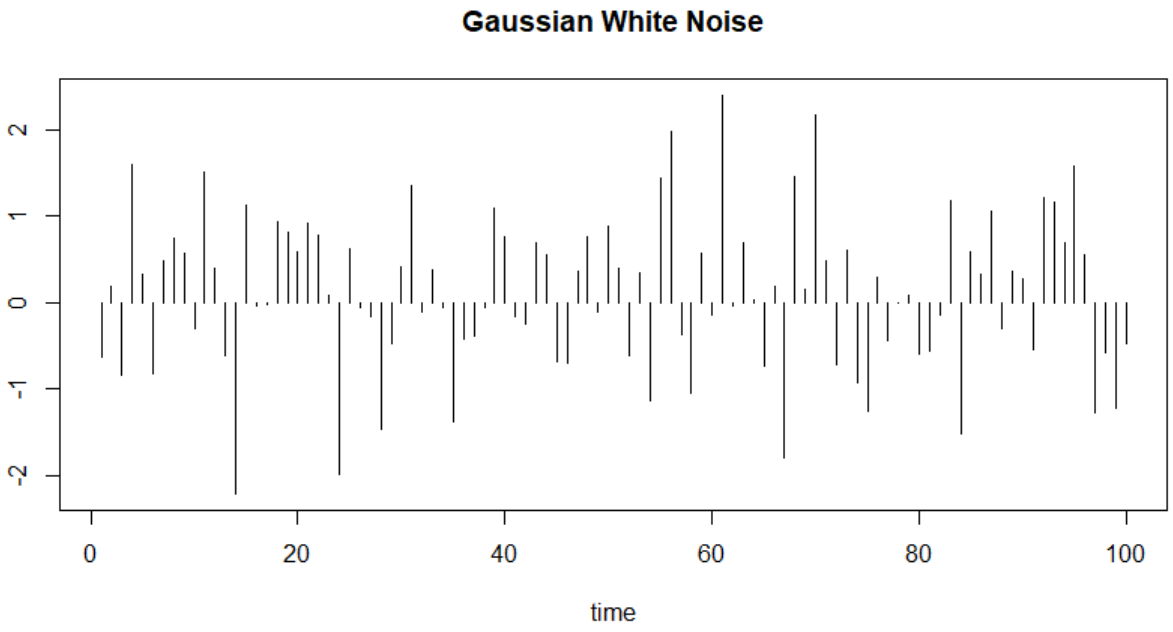
1.2.1.1 模拟的 Poisson 白噪声

```
> set.seed(1); x <- rpois(100, 1)
> plot(x, type="h", xlab="time", ylab="", main="Poisson White Noise")
```



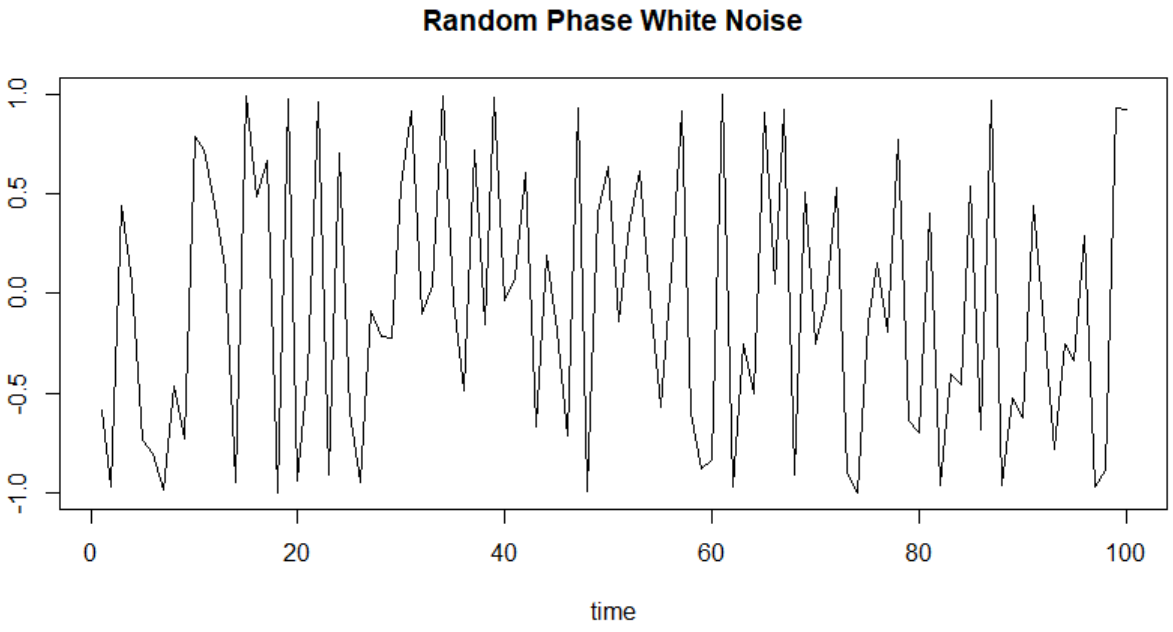
1.2.1.2 模拟的标准正态白噪声

```
> set.seed(1); x <- rnorm(100)
> plot(x, type="h", xlab="time", ylab="", main="Gaussian White Noise")
```



1.2.1.3 模拟的随机相位白噪声

```
> x <- cos(2*pi/12*(1:100) + runif(100, 0, 2*pi))
> plot(x, type="l", xlab="time", ylab="",
+      main="Random Phase White Noise")
```



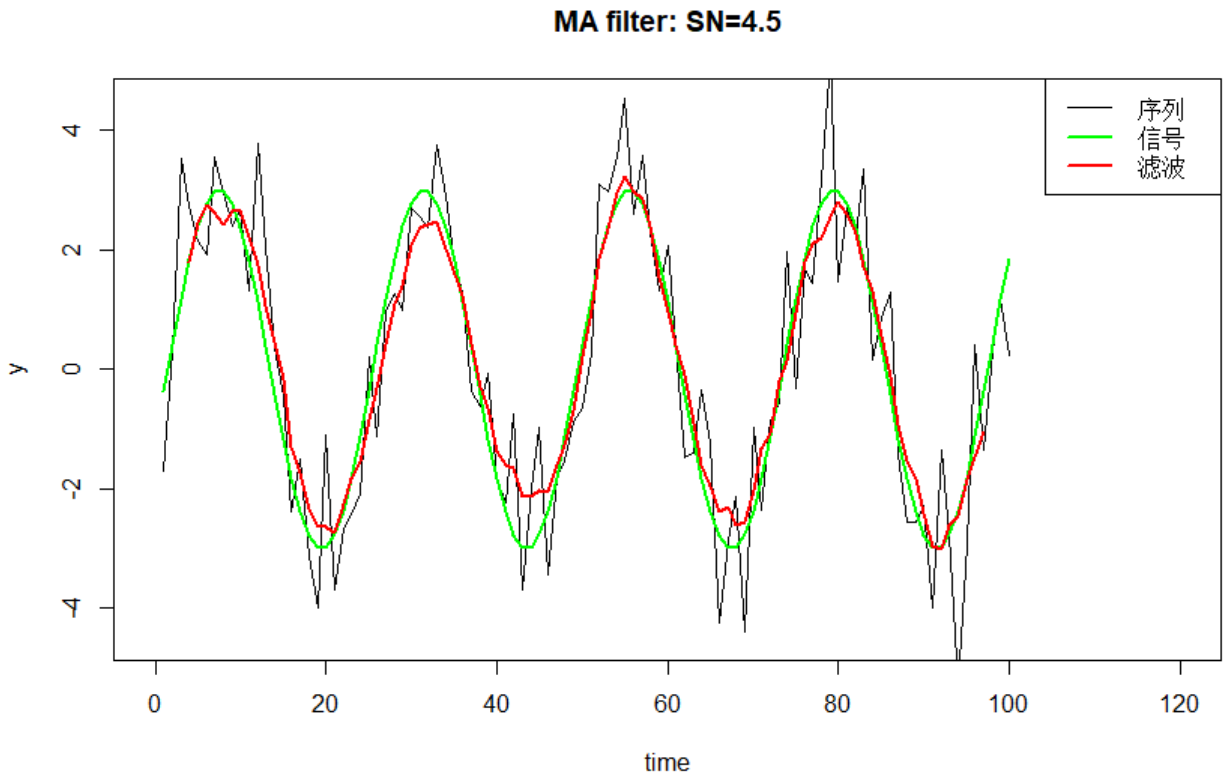
Chapter 2

线性平稳序列和线性滤波

2.1 有限运动平均

2.1.1 例: 余弦波信号的滤波

```
> demo.mafilt <- function(b=3, M=3){
+     n <- 100
+     ##om <- pi/7
+     om <- pi/12
+     sigma <- 1.0
+     eps <- rnorm(n, 0, sigma)
+     sn0 <- b^2 / (2*sigma^2)
+     tt <- seq(n)
+     u <- runif(1)
+     signal <- b * cos(om*tt + 2*pi*u)
+     y <- signal + eps
+     filt <- rep(1/(2*M+1), 2*M+1)
+     yf <- filter(y, filt, method="convolution")
+     rg <- range(c(y, yf))
+     sn <- round(b^2 / (2*sigma^2), 3)
+     plot(tt, y, main=paste("MA filter: SN=", sn, sep=""),
+          type="l", xlab='time', ylab='y',
+          xlim=c(0,120),
+          ylim=c(-b*1.5,b*1.5))
+     lines(tt, signal, col="green", lwd=2)
+     lines(tt, yf, col="red", lwd=2)
+     legend("topright", lty=c(1,1,1), lwd=c(1,2,2),
+            col=c("black", "green", "red"),
+            legend=c(" 序列", " 信号", " 滤波"))
+ }
> demo.mafilt()
```


MA filter: SN=4.5
序列
信号
滤波
y
time
4
2
0
-2
-4
0
20
40
60
80
100
120

Chapter 3

自回归模型及其平稳性

3.1 AR(1)

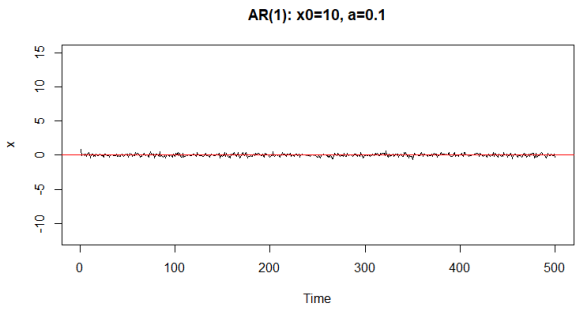
```
> ar1.gen <- function(n, a, sigma=1.0,
+                     plot.it=FALSE, n0=1000,
+                     x0=numeric(length(a))){
        +     n2 <- n0 + n
        +     eps <- rnorm(n2, 0, sigma)
        +     x2 <- filter(eps, a, method="recursive", side=1, init=x0)
        +     x <- x2[(n0+1):n2]
        +     x <- ts(x)
        + attr(x, "model") <- "AR(1)"
        + attr(x, "a") <- a
        + attr(x, "sigma") <- sigma
        + if(plot.it) plot(x)
        + x
        + }
> demo.ar1 <- function(){
        +     as <- c(0.1, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99, 1.0, 1.001)
        +     x0 <- 10
        +     n <- 500
        +     for(a in as){
          +         for(a1 in c(a, -a)){
                +         x <- ar1.gen(n=n, a=a1, sigma=0.2,
                +                      n0=0, x0=x0)
                +         plot(x, main=paste("AR(1): x0=10, a=", a1, sep=""),
                +              ylim=c(-12, 15))
                +         abline(h=0, col="red")
                +       }
        +     }
 + }
```

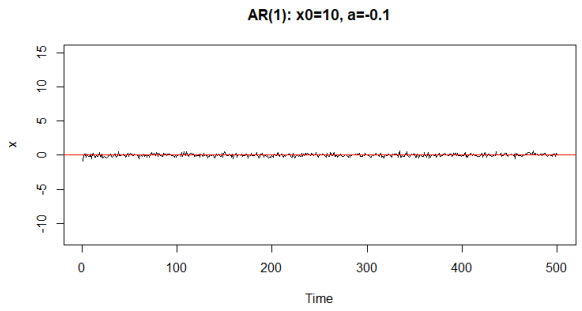
```
> set.seed(106)
> demo.ar1()
```

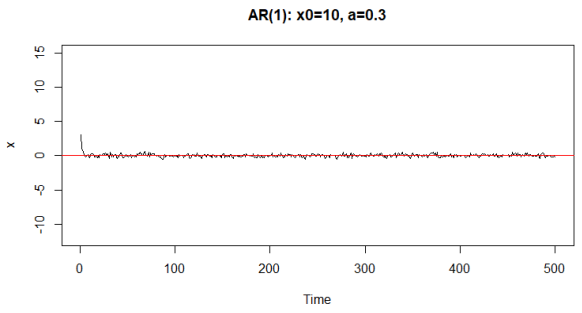
filter(x, filter, method = c("convolution", "recursive"), sides = 2, circular = FALSE, init)

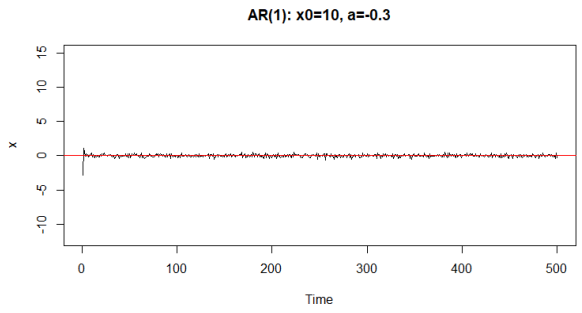
- x: 一元或多元时间序列。
- filter: 一个反向的时间顺序滤波器系数向量 (AR 或 MA 系数)。
- method: "convolution" 或者"recursive"(可以缩写)，如果是"convolution" 则是移动平均，如果是"recursive" 则是自回归。
- sides: 卷积过滤器。如果 sides=1 滤波器系数是对过去的价值观; 如果 sides=2 他们周围滞后 0 为中心的。在这种情况下，过滤器的长度应为奇数，但如果它是偶数，过滤器的更多的是向前的时间比落后。
- circular: 卷积过滤器。如果 TRUE，包装左右两端系列的过滤，否则假定缺少外部值（NA）。
- init: 只有递归滤波器。指定的时间序列的初始值，只是之前的起始值，反向时间顺序。默认的是一套零。

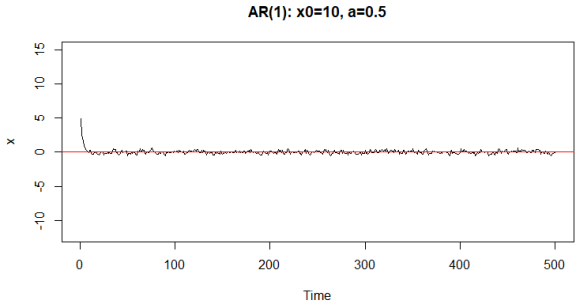
用 **attr(x, " 属性名")** 的格式读取或定义 x 的属性。可以让向量 x 额外地保存一个 theta 属性，这样的属性常常成为“元数据”(meta data)，比如，用来保存数据的说明、模拟数据的真实模型参数，等等。

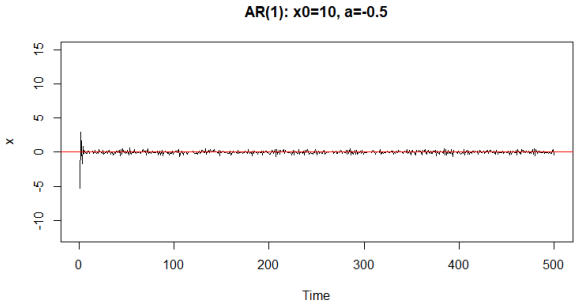


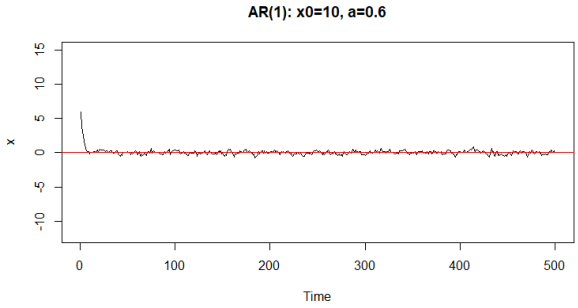


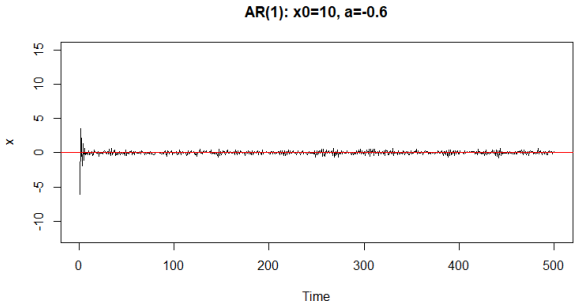


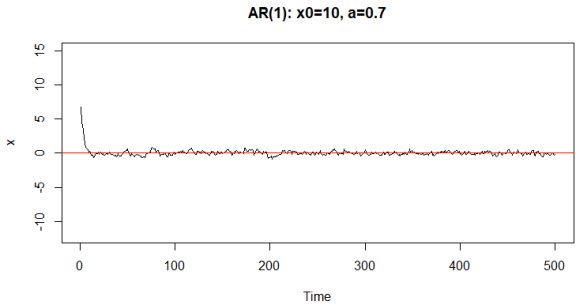


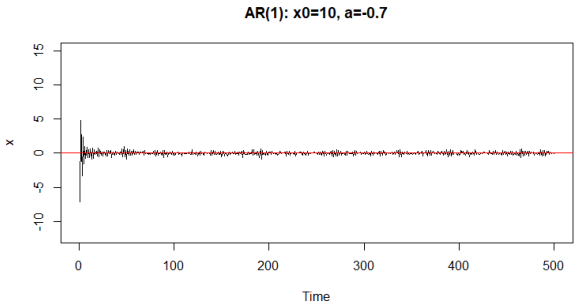

AR(1): x0=10, a=0.5
x
Time

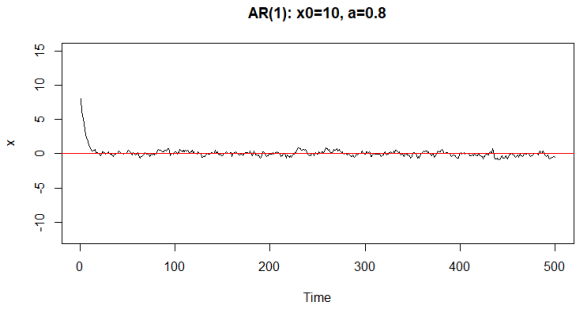

AR(1): x0=10, a=-0.5
x
Time

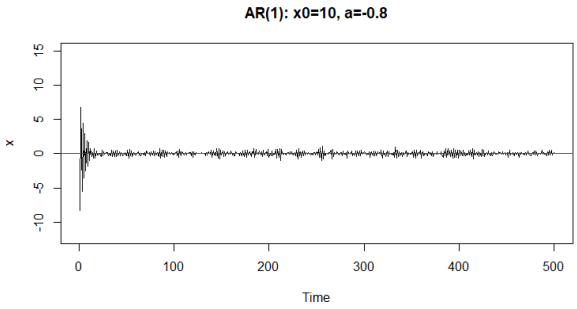

AR(1): x0=10, a=0.6
x
Time

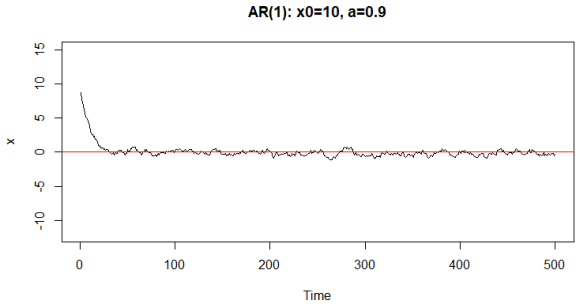

AR(1): x0=10, a=-0.6
x
Time

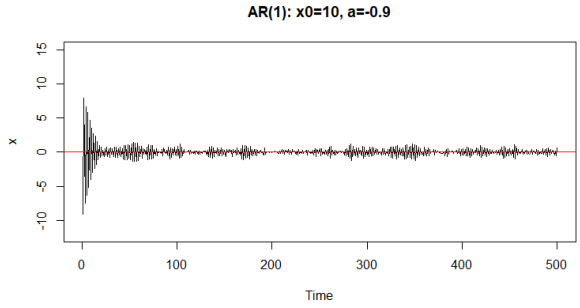

AR(1): x0=10, a=0.7
x
Time

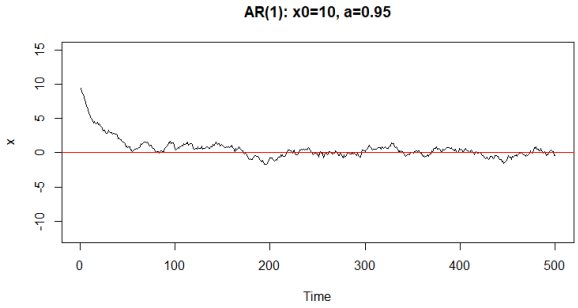

AR(1): x0=10, a=-0.7
x
Time

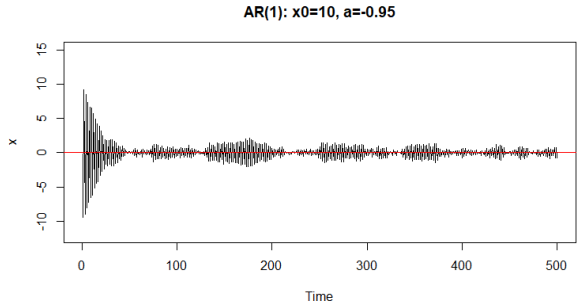

AR(1): x0=10, a=0.8
x
Time

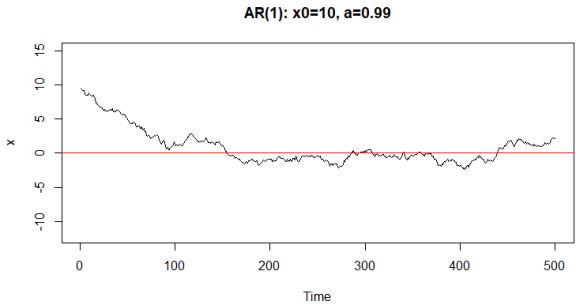

AR(1): x0=10, a=-0.8
x
Time

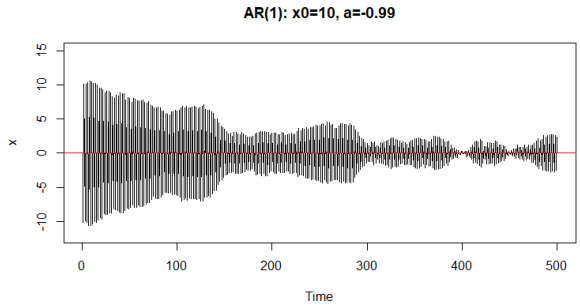

AR(1): x0=10, a=0.9
x
Time

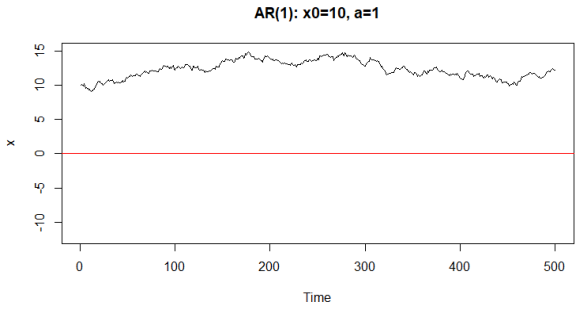

AR(1): x0=10, a=-0.9
x
Time

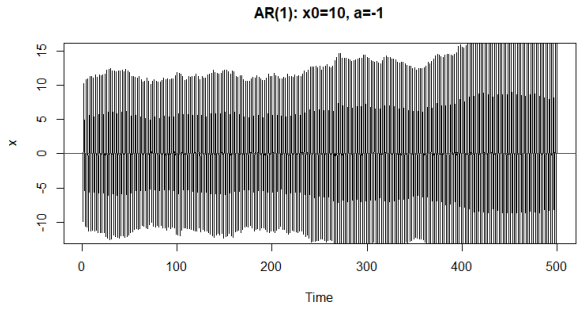

AR(1): x0=10, a=0.95
x
Time

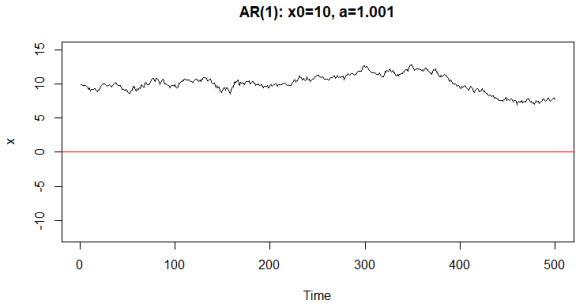

AR(1): x0=10, a=-0.95
x
Time

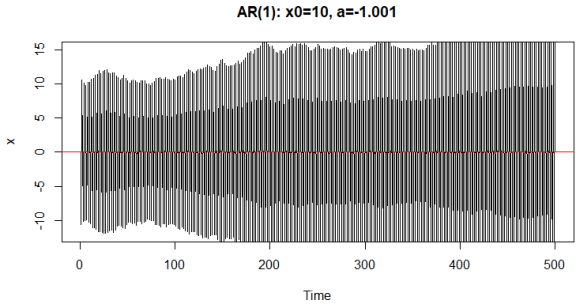

AR(1): x0=10, a=0.99
x
Time


AR(1): x0=10, a=-0.99
x
Time


AR(1): x0=10, a=1
x
Time


AR(1): x0=10, a=-1
x
Time


AR(1): x0=10, a=1.001
x
Time
-10
-5
0
5
10
15
0
100
200
300
400
500


AR(1): x0=10, a=-1.001
x
Time
-10
-5
0
5
10
15
0
100
200
300
400
500