R语言时间序列中文教程

2012

特别声明： R语言是免费语言，其代码不带任何质量保证，使用 R语言所产生的后果由使用者负全责。

前言

R语言是一种数据分析语言，它是科学的免费的数据分析语言，是凝聚了众多研究人员心血的成熟的使用范围广泛全面的语言，也是学习者能较快受益的语言。

在 R语言出现之前，数据分析的编程语言是 SAS，当时 SAS的功能比较有限。在贝尔实验室里，有一群科学家讨论提到，他们研究过程中需要用到数据分析软件。 SAS的局限也限制了他们的研究。于是他们想，我们贝尔实验室的研究历史要比 SAS长好几倍，技术力量也比 SAS强好几倍，且贝尔实验室里并不缺乏训练有素的专业编程人员， 那么，我们贝尔实验室为什么不自己编写数据分析语言， 来满足我们应用中所需要的特殊要求呢？

于是，贝尔实验室研究出了 S-PLUS语言。后来，新西兰奥克兰大学的两位教授非常青睐 S-PLUS的广泛性能。他们决定重新编写与 S-PLUS相似的语言，并且使之免费，提供给全世界所有相关研究人员使用。 于是，在这两位教授努力下， 一种叫做 R的语言在奥克兰大学诞生了。

R基本上是 S-PLUS的翻版，但 R是免费的语言，所有编程研究人员都可以对 R语言做出贡献， 且他们已经将大量研究成果写成了 R命令或脚本， 因而 R语言的功能比较强大，比较全面。

研究人员可免费使用 R语言，可通过阅读 R语言脚本源代码， 学习其他人的研究成果。笔者曾有幸在奥克兰大学受过几年熏陶， 曾经向一位统计系的老师提请教过一个数据模拟方面的问题。那位老师只用一行 R语句就解答了。 R语言的强大功能非常令人惊讶。

为了进一步推广 R语言，为了方便更多研究人员学习使用 R语言，我们收集了 R语言时间序列分析实例，以供大家了解和学习使用。当然，这是非常简单的模仿练习，具体操作是， 用复制粘贴把本材料中 R代码放入 R的编程环境；材料中蓝色背景的内容是相关代码和相应输出结果。经过反复模仿，学习者便能熟悉和学会。

需要提醒学习者的是：建议学习者安装了 R语言编程，再继续阅读本材料；执行 R命令时， 请删除命令的中文注解， 没使用过在命令中加入中文； 如果学习者是初次接触 R或者 Splus，建议先阅读 <<R语言样品比较应用举例 >>，如果学习者比较熟悉 R语言，还可以阅读优秀时间序列读物 Ecomometrics in R ，也可以上 QuickR 网站。

目录

R 语言时间序列分析 1
前言 1
目录 2
1.运用 R 语言研究 JJ数据 3
2.运用 R 语言研究空气污染 19
3.运用 R 语言研究自动回归移动平均集成模型 22
4.运用 R 语言研究全球变暖理论 30
5.运用 R 语言研究非独立误差与线性回归 32
6.运用 R 语言研究估计波动的频率 34
7.运用 R 语言研究厄尔尼诺频率 38
8.运用 R 语言研究太阳黑子周期频率 39

火龙果·整理
uml.org.cn

目录

R 语言时间序列分析 1
前言 1
目录 2
1.运用 R 语言研究 JJ数据 3
2.运用 R 语言研究空气污染 19
3.运用 R 语言研究自动回归移动平均集成模型 22
4.运用 R 语言研究全球变暖理论 30
5.运用 R 语言研究非独立误差与线性回归 32
6.运用 R 语言研究估计波动的频率 34
7.运用 R 语言研究厄尔尼诺频率 38
8.运用 R 语言研究太阳黑子周期频率 39

如果使用 matlab，你会认为 jj 是一个 84 行 1 列的向量，但实际上不是这样。 jj 有次序，有长度，但没维度， R称这些对象为向量，要小心区别。
在 R里，矩阵有维度，但向量没维度。这都是程序语言的一些概念。

```
jj[1]      #          the first element          列中第一个数据

 [1] 0.71

jj[84]          # the last element          列中最后一个数据

 [1] 11.61

jj[1:4]    #          the first 4 elements          列中第一至第四个数据

 [1] 0.71 0.63 0.85 0.44

jj[-(1:80)] #          everything EXCEPT the first 80 elements          列中除第 80
个以外的所有数据
 [1] 16.20 14.67 16.02 11.61

length(jj)  #          the number of elements          有多少个数据

 [1] 84

dim(jj)    #          but no dimensions ...          但没维度

 NULL

nrow(jj)          # ... no rows          没行

 NULL

ncol(jj)          # ... and no columns          没列

 NULL

#如果你要把 jj 转变为一个向量，执行如下操作后，维度为 84 行 1 列。

jj = as.matrix(jj)
dim(jj)
 [1] 84        1
```

然后把 jj 转变为一个时间序列对象。

火龙果·整理
uml.org.cn

如果使用 matlab，你会认为 jj 是一个 84 行 1 列的向量，但实际上不是这样。 jj 有次序，有长度，但没维度， R称这些对象为向量，要小心区别。
在 R里，矩阵有维度，但向量没维度。这都是程序语言的一些概念。

```
jj[1]      #          the first element          列中第一个数据

 [1] 0.71

jj[84]          # the last element          列中最后一个数据

 [1] 11.61

jj[1:4]    #          the first 4 elements          列中第一至第四个数据

 [1] 0.71 0.63 0.85 0.44

jj[-(1:80)] #          everything EXCEPT the first 80 elements          列中除第 80
个以外的所有数据
 [1] 16.20 14.67 16.02 11.61

length(jj)  #          the number of elements          有多少个数据

 [1] 84

dim(jj)    #          but no dimensions ...          但没维度

 NULL

nrow(jj)          # ... no rows          没行

 NULL

ncol(jj)          # ... and no columns          没列

 NULL

#如果你要把 jj 转变为一个向量，执行如下操作后，维度为 84 行 1 列。

jj = as.matrix(jj)
dim(jj)
 [1] 84        1
```

然后把 jj 转变为一个时间序列对象。

火龙果·整理
uml.org.cn

如果使用 matlab，你会认为 jj 是一个 84 行 1 列的向量，但实际上不是这样。 jj 有次序，有长度，但没维度， R称这些对象为向量，要小心区别。
在 R里，矩阵有维度，但向量没维度。这都是程序语言的一些概念。

```
jj[1]      #          the first element          列中第一个数据

 [1] 0.71

jj[84]          # the last element          列中最后一个数据

 [1] 11.61

jj[1:4]    #          the first 4 elements          列中第一至第四个数据

 [1] 0.71 0.63 0.85 0.44

jj[-(1:80)] #          everything EXCEPT the first 80 elements          列中除第 80
个以外的所有数据
 [1] 16.20 14.67 16.02 11.61

length(jj)  #          the number of elements          有多少个数据

 [1] 84

dim(jj)    #          but no dimensions ...          但没维度

 NULL

nrow(jj)          # ... no rows          没行

 NULL

ncol(jj)          # ... and no columns          没列

 NULL

#如果你要把 jj 转变为一个向量，执行如下操作后，维度为 84 行 1 列。

jj = as.matrix(jj)
dim(jj)
 [1] 84        1
```

然后把 jj 转变为一个时间序列对象。

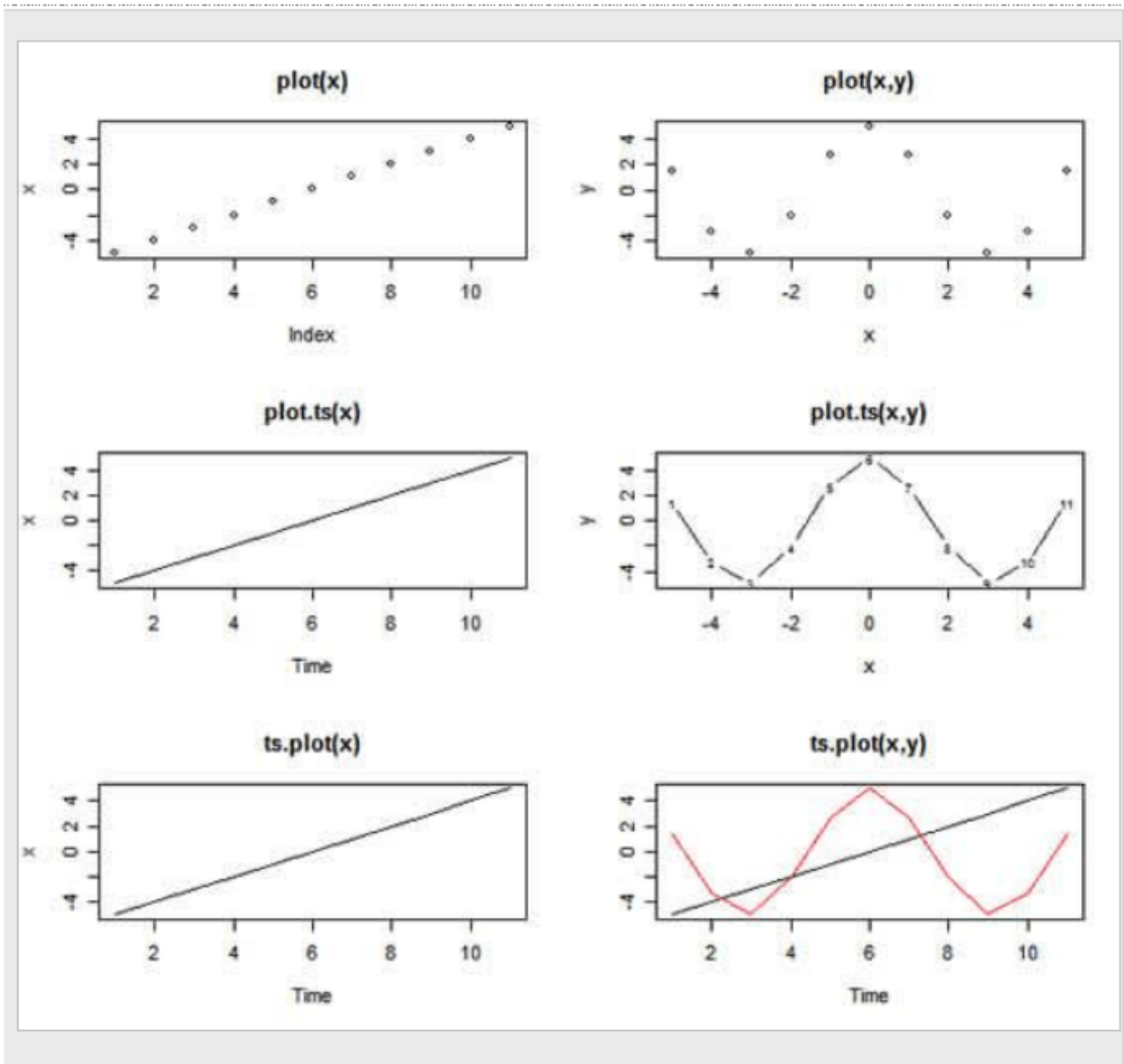
火龙果·整理
uml.org.cn

如果使用 matlab，你会认为 jj 是一个 84 行 1 列的向量，但实际上不是这样。 jj 有次序，有长度，但没维度， R称这些对象为向量，要小心区别。
在 R里，矩阵有维度，但向量没维度。这都是程序语言的一些概念。

```
jj[1]      #          the first element          列中第一个数据

 [1] 0.71

jj[84]          # the last element          列中最后一个数据

 [1] 11.61

jj[1:4]    #          the first 4 elements          列中第一至第四个数据

 [1] 0.71 0.63 0.85 0.44

jj[-(1:80)] #          everything EXCEPT the first 80 elements          列中除第 80
个以外的所有数据
 [1] 16.20 14.67 16.02 11.61

length(jj)  #          the number of elements          有多少个数据

 [1] 84

dim(jj)    #          but no dimensions ...          但没维度

 NULL

nrow(jj)          # ... no rows          没行

 NULL

ncol(jj)          # ... and no columns          没列

 NULL

#如果你要把 jj 转变为一个向量，执行如下操作后，维度为 84 行 1 列。

jj = as.matrix(jj)
dim(jj)
 [1] 84        1
```

然后把 jj 转变为一个时间序列对象。

```
ts.plot(x, main="ts.plot(x)")
ts.plot(ts(x), ts(y), col=1:2, main="ts.plot(x,y)") # note- x and y are ts objects
#--- the help files [? and help() are the same]:
?plot.ts
help(ts.plot)
?par # might as well skim the graphical parameters help file while you're here
```



从窗口中的显示可以看出，如果数据是时间序列对象，使用 plot() 命令就足够了；如果数据是平常序列，使用 plot.ts() 也可以做时间绘图。

不过，把 jj 数据放在一张图上，数据会随着时间的变化上上下下跳动，能从整体上反应上升或者下降的趋势。上文中用红色光滑的曲线代表上升的趋势，简单明了。

这需要将过滤和光滑的技巧使用在 jj 数据上。在这里，我们用对称的移动平均值来达到过滤和光滑的目的。

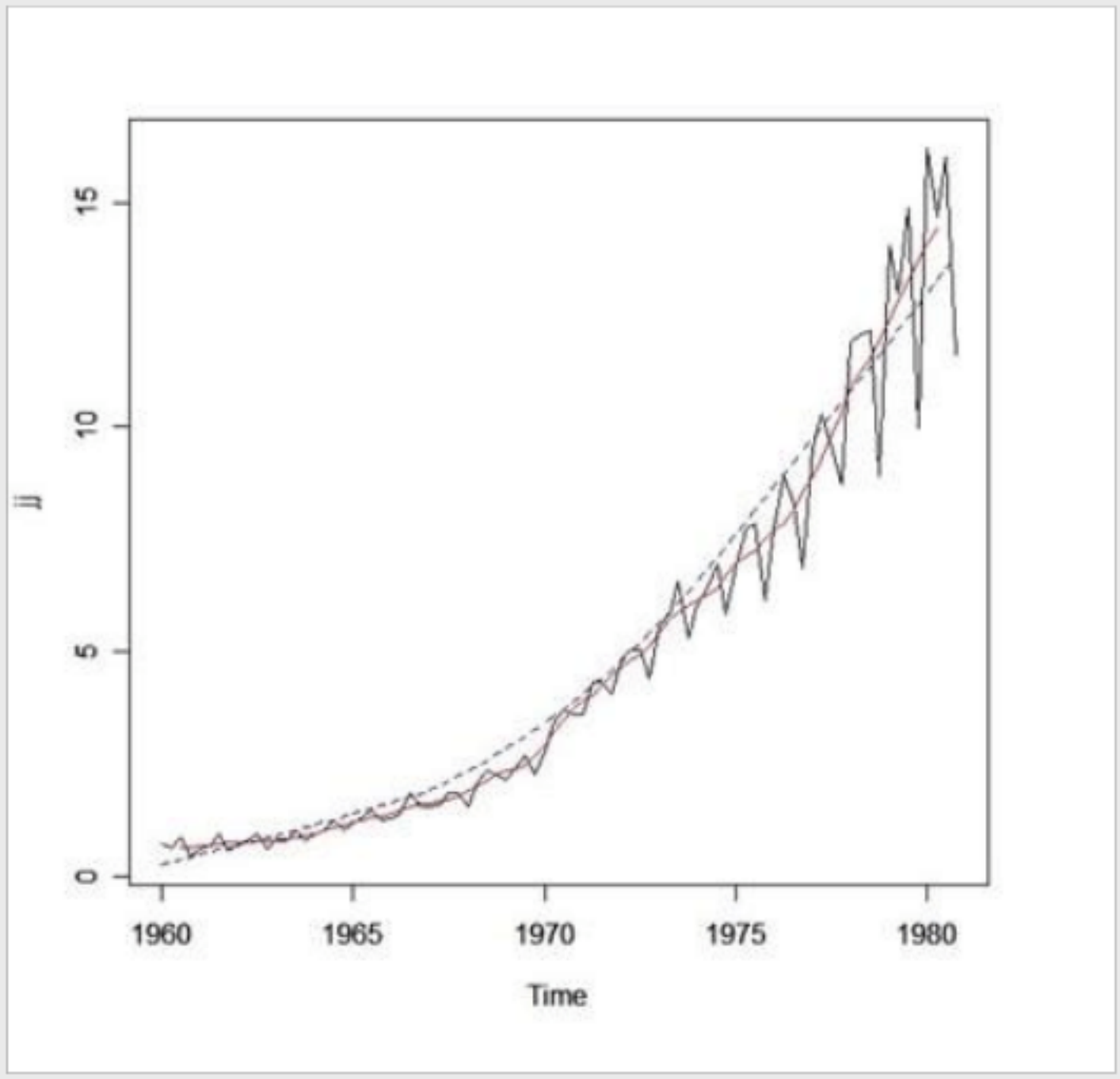
下面使用公式：

fjj(t) = ? jj(t-2) + ? jj(t-1) + ? jj(t) + ? jj(t+1) + ? jj(t+2)

除此之外， lowess 的过滤平滑技巧（蓝色曲线）也要使用在 jj 数据中。具体操作如下图：

```
k = c(.5,1,1,1,.5)          #                    k is the vector of weights                    用于对称移动平均的系数

(k = k/sum(k))

  [1] 0.125 0.250 0.250 0.250 0.125

fjj = filter(jj, sides=2, k)  #                    ?filter for help [but you knew that already]                    使用对称移动平均

plot(jj)

lines(fjj, col="red")        #                    adds a line to the existing plot                    称移动平均的绘图

lines(lowess(jj), col="blue", lty="dashed")#lowess                    的绘图
```

操作后，窗口会显示下面结果：



看完 jj 数据，我们就需要开始具体分析。

第一步，我们把所有 jj 数据都取 log 值。第二步，我们把 log 值做差，即使用 log 值数列中第二值减去第一值，第三值减去第二值，第四值减去第三值等等。如果做差处理前数列里有 n 个数值，处理后的结果中将有 n-1 个数值。

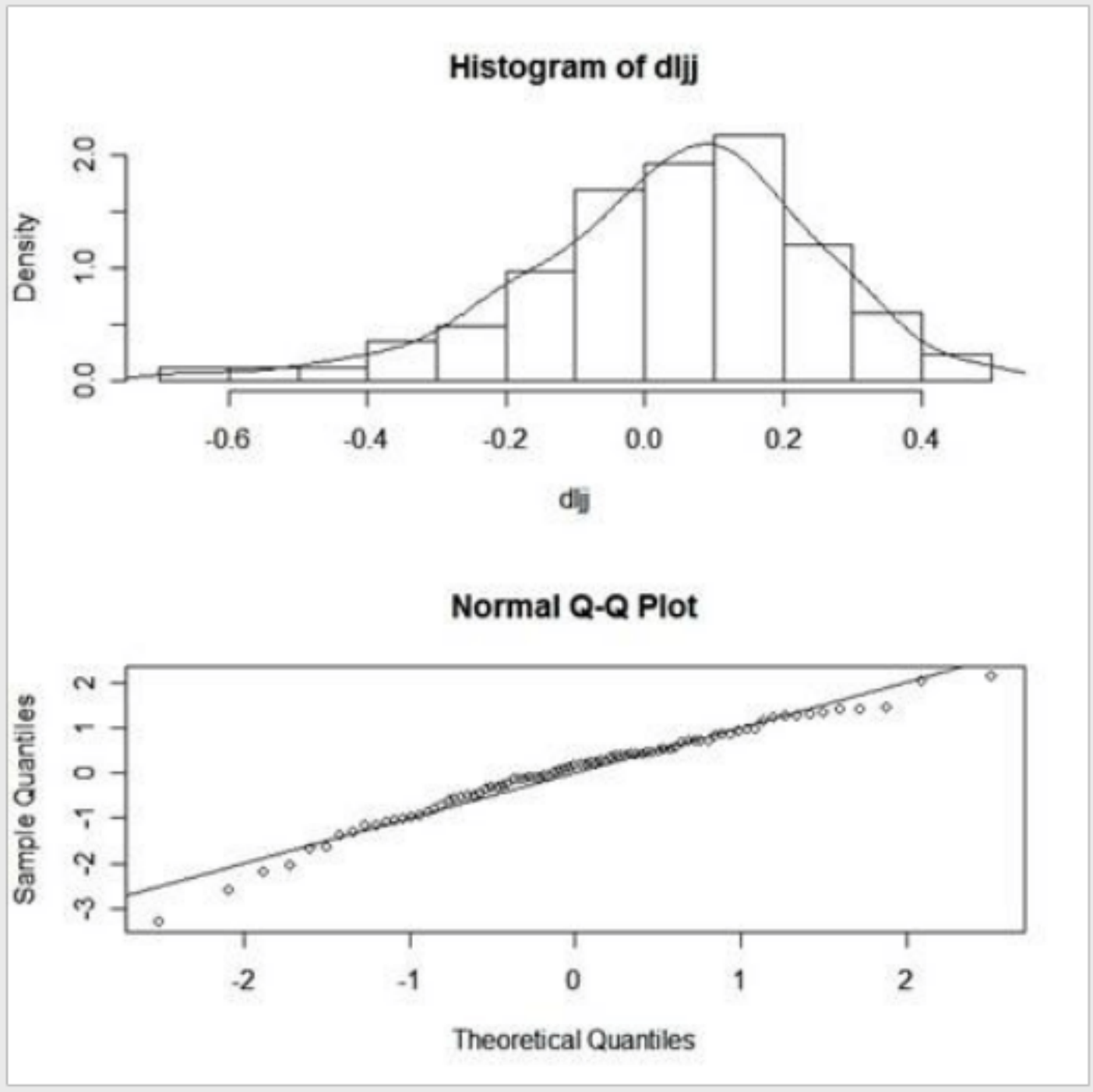
```
dljj = diff(log(jj))                    #  difference the logged data          做 log 和差的处理
plot(dljj)                              #  plot it if you haven't already      对结果制图
shapiro.test(dljj)                      #  test for normality                  测试结果的正态分布的性质

Shapiro-Wilk normality test

data:  dljj

W = 0.9725, p-value = 0.07211
```

处理完毕以上两步，我们接下来就要将柱形分布图和 QQ图放在一起。这两个图的本质仍旧在于测试数据正态分布的性质。数据正态分布的性质是整个统计学构架坚实的基础，如果这个性质的存在比较可信、通过了所有测试，统计分析中得出的结论就比较可信、就通得过所有测试。当然如果这个性质在数据中不存在，我们需要用其它的技巧来处理。详细的，参看 R语言样品比较应用的实例。

以上操作，在窗口中有如下显示：

```
par(mfrow=c(2,1))       #  set up the graphics          设置为两图的输出
hist(dljj, prob=TRUE, 12)   #  histogram       柱形分布图
lines(density(dljj))     #  smooth it - ?density for details          柱形分布图的曲线
qqnorm(dljj)            #  normal Q-Q plot  QQ 图
qqline(dljj)            #  add a line          在 QQ 图 上 加 直线
```

经过测试数据后，窗口会有如下显示：



在实践操作中，时间序列数据存在着前后关系。例如，今天股票的价格很有可能决定明天股票的价格。明天的温度取决于今天的气温。做天气预报的具体操作方法，是使用已经存在的天气历史记录，比如说今天的气温，昨天的气温，前天的气温等等，来预测明天的气温。当然，在进行预测之前，我们一定要看清时间序列数据中的前后关系结构，清楚哪一个特定的历史数据可以精确预测未来的数据。在这里，我们使用被 log 和求差后的 dljj 数据，来介绍分析时间序列数据前后关系结构的具体技巧。

在预测的实际应用中，我们总希望用历史数据来预测即将要产生的数据。事实上，已产生的数据相对于即将产生的数据，中间存在着一定的延迟，也就是 lag. 比方说在某天凌晨 12 点开始记录室内温度，每小时记一次，一共连续记录了 10 个小时。凌晨 12 点的数据和凌晨 3 点的数据之间就存在着延迟。12 点的数据比 3 点的早了 3 个小时，可记作 lag3. 3 点的数据比 12 点的晚了 3 个小时，可记作 lead3.

我们下面来介绍关联性。例如，冷饮的销量与天气温度存在关联性。温度越高冷饮销量就越高，温度越低冷饮销量也越低。这种关联性称为正面关联性。又如，人的体重和跑步速度也存在关联性。不过，人的体重越重，跑步速度就会越慢，体重越低，相对来讲，速度就会越快。这种关联性称为负面关联性。

下面我们回到预测应用上。如果现在收集的数据与将来的数据之间存在着正面或者是负面的关联性，我们就可以用现在收集的数据来预测未来的数据。因此找到现在收集到的数据与未来数据之间的关联性是最关键的。找到这种关联性的具体技巧被称作延迟图表，也就是 lag.plot.

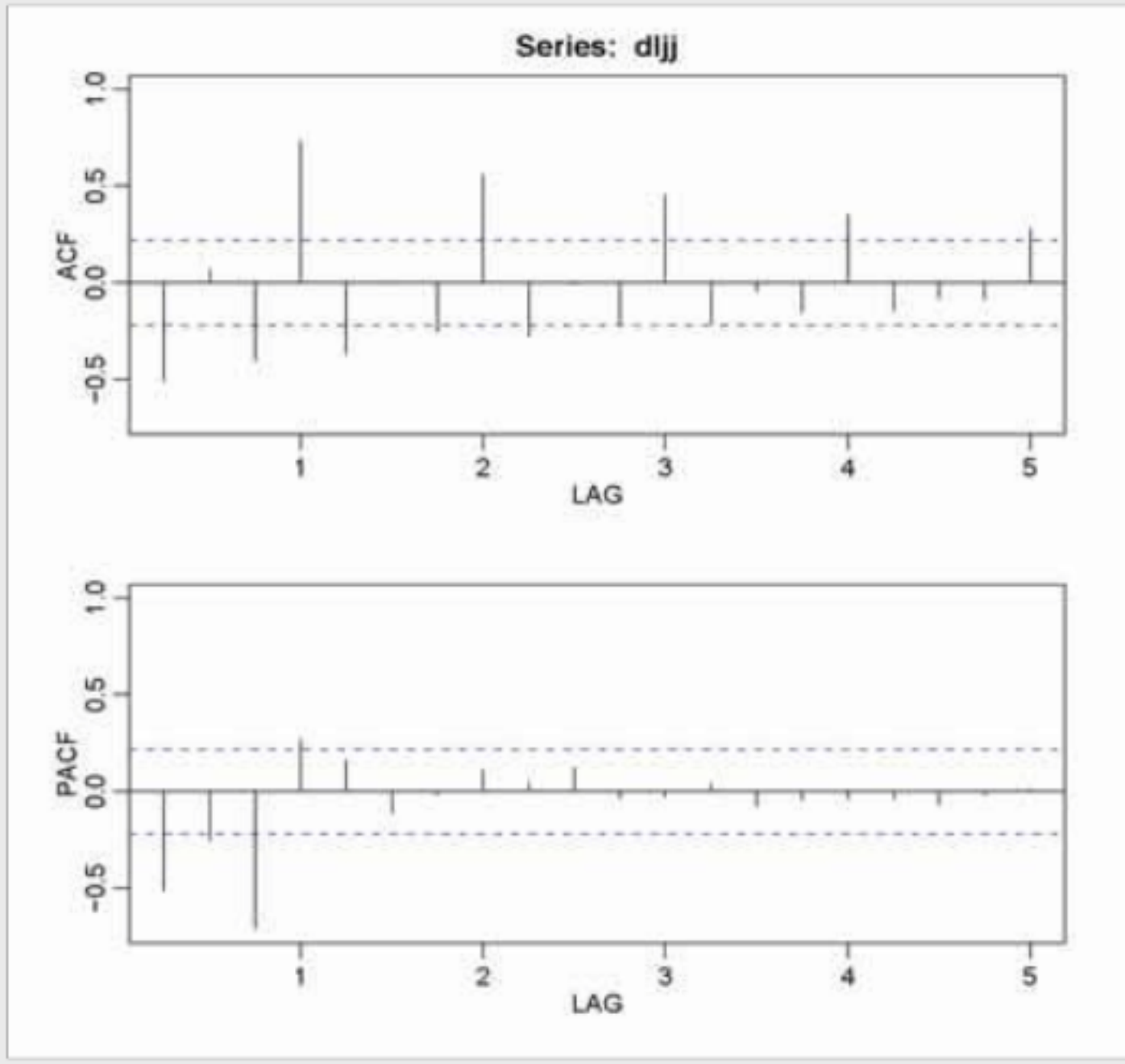
在实践操作中，时间序列数据存在着前后关系。例如，今天股票的价格很有可能决定明天股票的价格。明天的温度取决于今天的气温。做天气预报的具体操作方法，是使用已经存在的天气历史记录，比如说今天的气温，昨天的气温，前天的气温等等，来预测明天的气温。当然，在进行预测之前，我们一定要看清时间序列数据中的前后关系结构，清楚哪一个特定的历史数据可以精确预测未来的数据。在这里，我们使用被 log 和求差后的 dljj 数据，来介绍分析时间序列数据前后关系结构的具体技巧。

在预测的实际应用中，我们总希望用历史数据来预测即将要产生的数据。事实上，已产生的数据相对于即将产生的数据，中间存在着一定的延迟，也就是 lag. 比方说在某天凌晨 12 点开始记录室内温度，每小时记一次，一共连续记录了 10 个小时。凌晨 12 点的数据和凌晨 3 点的数据之间就存在着延迟。12 点的数据比 3 点的早了 3 个小时，可记作 lag3. 3 点的数据比 12 点的晚了 3 个小时，可记作 lead3.

我们下面来介绍关联性。例如，冷饮的销量与天气温度存在关联性。温度越高冷饮销量就越高，温度越低冷饮销量也越低。这种关联性称为正面关联性。又如，人的体重和跑步速度也存在关联性。不过，人的体重越重，跑步速度就会越慢，体重越低，相对来讲，速度就会越快。这种关联性称为负面关联性。

下面我们回到预测应用上。如果现在收集的数据与将来的数据之间存在着正面或者是负面的关联性，我们就可以用现在收集的数据来预测未来的数据。因此找到现在收集到的数据与未来数据之间的关联性是最关键的。找到这种关联性的具体技巧被称作延迟图表，也就是 lag.plot.

```
acf(dljj, 20)  # ACF to lag 20 - no graph shown... keep reading
pacf(dljj, 20)  # PACF to lag 20 - no graph shown... keep reading
# !!NOTE!! acf2 on the line below is NOT available in R... details follow the graph below
acf2(dljj)  # this is what you'll see below
```



在上图中，ACF和 PACF横坐标的标记是 1,2,3,4,5. 但因为数据是季度性的，每年有 4 个季度所以 1,2,3,4,5 的标记代表的 4,8,12,16,20 的延迟。

当然，如果我们不喜欢上面横坐标的标记，也可以将 dljj 更改为 ts(dljj, freq=1); 也就是说 acf(ts(dljj, freq=1), 20)。因为在上面 ACF图表中横坐标 1 代表的是延迟 4，横坐标 2 代表的是延迟 6，其相应的竖线代表的就是延迟 4 和 8 的正面关系。

接下来，下面我们介绍结构拆析。在前面 R代码中，我们曾将所有 jj 数据进行了 lag 变型。在变型后的数据中，存在着上升趋势，季节的影响和每一时间点产生的随机的误差。根据这一数据图，我们能够把趋势、季节和误差从变型后的 jj 数据中拆析出来，分别研究，或者分别进行绘图，以便于单独检查。

下面等式代表将要使用的数学模型：

Log(jj)= 趋势+季节+误差

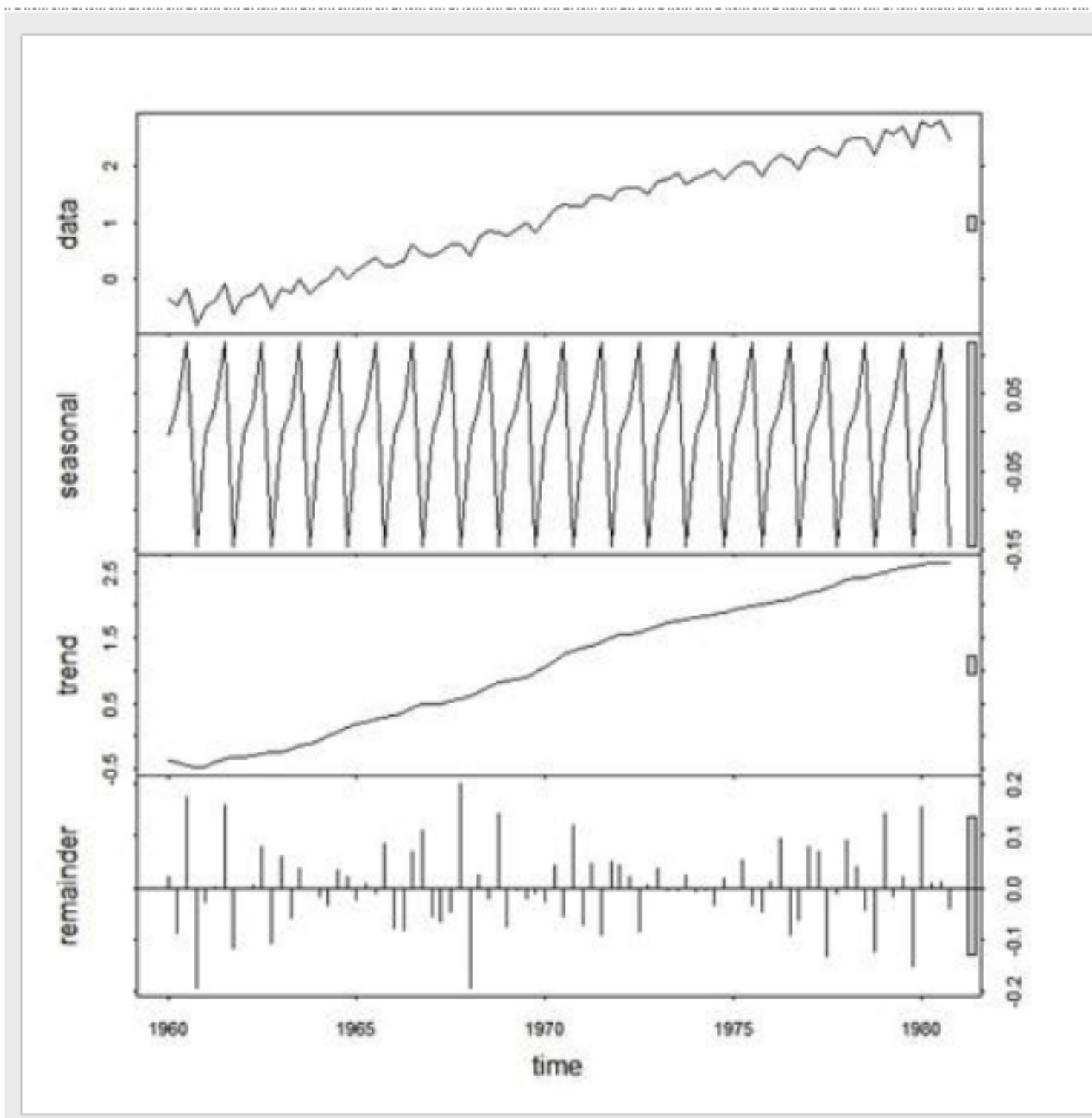
log(jj) = trend + season + error

结构拆析的 R 命令是 stl(), 下面语句中 stl 命令中输入的是 lag 变型后的 jj 数据。其中的“ per ”输入指的是使用季节循环来进行拆析。 stl 语句在这里生成了一个叫 dog 的 R 物件，然后 Plot 语句将 dog 物件进行绘图。

具体操作如下图 ”

```
plot(dog <- stl(log(jj), "per"))
```

窗口会出现下面所示：



上图中有四行 R 的绘图，其中第一行代表原来的 log（jj ）的数据。此数据可以看到总体的上升趋势还存在着一定季节循环性的变化。 第二行绘图代表拆析后季节循环的作用。第三行绘图代表拆析后将季节循环作用清除剩余的上升趋势， 此数据清楚地看到那种循环性变化已经不存在， 剩余的只是趋势。 第四行绘图代表将季节循环作用和总体的趋势从数据中清除后所剩余的随机产生的误差。

如果我们需要对数据的误差进行一些常规检测，例如进行正态分布检测，绘制 QQ 图，还有绘制柱形图。我们所需要的具体误差数据被存在叫做 dog$time.series[,3] 的数列里。即叫 dog 的物件中有个叫 time.series 的数据矩阵， 误差就被存储在这个数据矩阵的第三列里。 $指调取 dog 物件中的 time.series 数据矩阵。 [,3] 指数据矩阵中第

三列。如果要对这一数列的误差值进行 ACF的分析，只需要执行命令 acf(dog$time.series[,3]) 。

再接下来，我们对 log 变型后的 jj 数据进行线性回归模型分析。与上面结构拆析不同的是，我们在这里使用四个季度来量化季节循环对数据的影响。 一年中有四个季度，也是我们所使用数据所代表的。这个 jj 数据是某一家公司的季度收入数据，从上面绘图中我们就可以看到， 每一年第三季度就会出现一个收入高峰， 随之而来第四季度收入就会跌入低谷。 然后在一季度和二季度收入又会逐渐上升。 这也就是说， 每一季度对这家公司收入的影响都是不一样的。

具体考虑到这种季度之间的不同，我们可使用如下数学模型：

$$\log(jj)= \beta*time + \alpha_1*Q1 + \alpha_2*Q2 + \alpha_3*Q3 + \alpha_4*Q4 + \varepsilon$$

这个数学模型的意思是：

log(jj)= 趋势*时间+一季度的影响 *一季度 +二季度的影响 *二季度 +三季度的影响 *三季度 +四季度的影响 *四季度 +误差

上面的模型 β 代表的就是总体上升趋势， α_1 α_2 α_3 α_4代表的是四个季度的影响。有一个非常有趣的问题， 上面模型是把所有四个季度的趋势都加在了一起， 其结果却是某单一季度的收入。 四个季度的和如何能够与一个季度相等问题就出在 Q1Q2Q3Q4上。因为 Q被我们称作指示性函数。 函数的意思就是数据进， 数据出， 也就是说把一个数据输入到一个函数中，那个函数就会输出一个结果。

以上面的 Q1函数为例，Q1只能输出两种结果， 1 和 0. Q1所需要的输入是四种 1, 2, 3, 4, 代表四个季度。 把 1 输入到 Q1函数中时，Q1函数输出的结果为 1，当把 2,3,4 输入 Q1函数时， Q1函数输出的结果为 0.

与 Q1函数类似， Q2函数的输入也是 1,2,3,4 ， 但只有输入为 2 时，Q2函数的输出才为 1，当输入为 1,3,4 时，Q2函数的输出为 0. Q3函数输入为 1,2,3,4 ，只有当输入为 3 时，输出为 1，输入其他数据时， 输出为 0.Q4 函数的输入为 1,2,3,4 ，只有当输入为 4 时，输出为 1，其他数据时，输出为 0.

我们再回到上面的模型，当一个数据是从第一季度中记录下来的， Q1给出数值 1，Q2给出数值 0，Q3给出数值 0，Q4给出的数值 0。因为这时 Q2，Q3，Q4都是 0，二季度，三季度， 四季度的影响被 0 相乘后也变成了 0. 所以在第一季度 Q1为 1，而其他的为 0. 我们就只考虑了一季度的影响，其他季度的影响不存在。同理，当季度为二、三、四时也有类似结果。

下面是建立这个线性模型的 R 语句，只有头三行是用来生成线性模型的， 第四条语句 summary()用来输出模型参数数值。

具体操作以及显示如下：

```
Q = factor(rep(1:4,21))     #                    make (Q)uarter factors [that's
repeat 1,2,3,4, 21 times]
trend     = time(jj)-1970              # not    necessary     to   "center" time,          but the
results look nicer
reg   =  lm(log(jj)~0+trend+Q,              na.action=NULL)  #          run   the   regression
without an intercept
#-- the na.action statement is to retain time series attributes
summary(reg)
    Call:
lm(formula = log(jj) ~ 0 + trend + Q, na.action = NULL)


Residuals:
    Min                 1Q   Median                3Q      Max
-0.29318 -0.09062 -0.01180                 0.08460  0.27644


Coefficients:
     Estimate Std. Error t value Pr(>|t|)
trend 0.167172   0.002259   74.00   <2e-16 ***
Q1    1.052793   0.027359   38.48   <2e-16 ***
Q2    1.080916   0.027365   39.50   <2e-16 ***
Q3    1.151024   0.027383   42.03   <2e-16 ***
Q4    0.882266   0.027412   32.19   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 0.1254 on 79 degrees of freedom
Multiple R-squared: 0.9935,     Adjusted R-squared: 0.9931
F-statistic:  2407 on 5 and 79 DF,  p-value: < 2.2e-16
```

上面语句第一行用来生成所需要的 Q函数。第二行用来生成线性回归模型中的时间输入，然后存储于叫做 trend 的数列中。第三行语句是用 lm() 命令来建立线性模型。在 R 语言中， ~用来分隔进行预测的变量和被预测的变量。

在上面语句第三行中， log （jj ）是被预测的变量，它被放置在 ~的左边。而且 0, trend， 和 Q函数都是用来进行预测的变量。 它们被放在 ~右边。其目的就是要用 0TREND 和 Q function 来预测 Log（jj ）。

现在，我们复习一下什么叫做线性。 线性是在坐标平面上的一条直线。 坐标平面上有两个数轴， 横轴叫做 x 轴，竖轴叫做 y 轴。在平面上画一条直， 可使用截距和斜率定义它。如果这条直线通过原点，其截距为 0. 在~右边的 0 是我们模型中的截距，也就是说我们模型中的直线通过原点。 如果我们决定使用有截距的线性模型， 即模型中的直线不通过原点，就要把第三条 R语句中的 0 改为 1. 这样，R语言就可以将截距计算出来。 执行 summary（）命令后，我们就可得到具体模型参数的数值。模型中的趋势，即 β 估计值为 0.167172，一季度的影响记为 a_1，估计值为 1.052793，第二季度的影响记为 a_2，估计值为 1.080916，第三季度的影响记为 a_3，估计值为 1.151024 ，第四季度的影响记为 a_4，估计值为 0.882266 。

随后，我们将所有估计值插入上面模型中，可得到下面等式：

Log(jj)=0.167172* 时间 +1.052793*Q1+ 1.080916*Q2+ 1.151024 *Q3+0.882266*Q4+误差

然后，我们就可以进行预测。比如说，在第 200 个时间所处的季度为 4，其预测值就可被计算为：

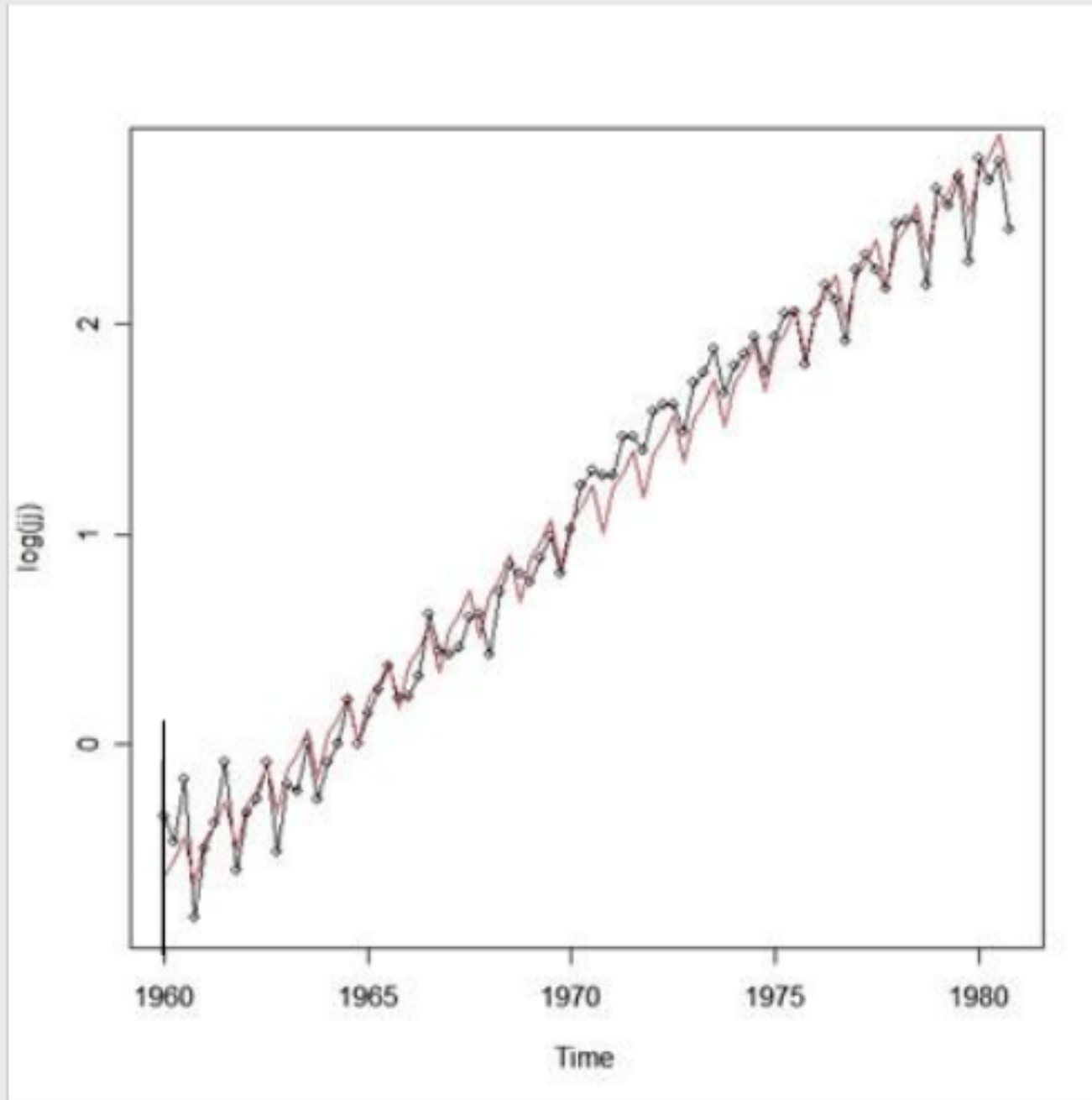
Log(jj)= 0.167172*200+1.052793*0+ 1.080916*0+ 1.151024 *0 +0.882266*1=Now check out what happened. Look at a plot of the observations and their fitted values:

在上面， 我们已经知道如何使用计算出来的模型对某一时间点进行预测。 但是，我们的预测是否符合在实际中观察到的时间序列数据， 我们需要将实际当中观察到的数据与模型计算出来的数据放在一起，绘成图表进行比较。

接下来要进行的工作是，将预测的数据和实观察的数据进行比较。下面第一条语句是对实际观察到的 log(jj) 数据进行绘图。第二行语句是对计算出的数据进行绘图，其中 fitted(reg) 是使用我们已建立好的模型来计算预期值。

```
plot(log(jj), type="o") # the data in black with little dots
lines(fitted(reg), col=2) # the fitted values in bloody red -
or use lines(reg$fitted, col=2)
```

输入上面两个命令后， 窗口就会显示如下图表。 在图表中， 黑线代表实际中观察到的 log(jj) 数据。红线代表由模型计算出来的 log(jj) 数据。它们看上去非常相似，但区别在于模型中计算出来的数值没误差。



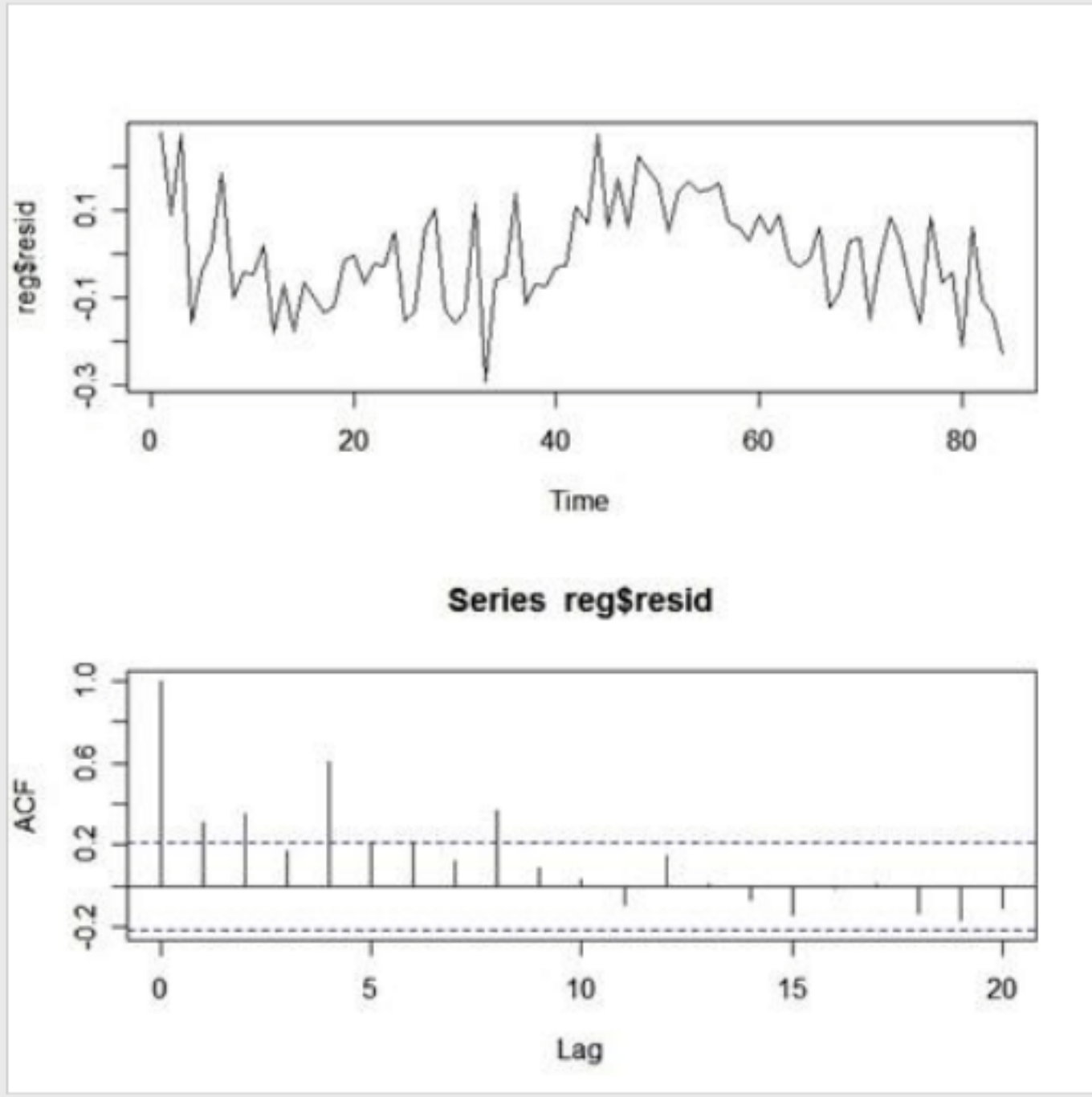
... and a plot of the residuals and the ACF of the residuals:

在对 log(jj) 建立模型时，具体的误差也被估计了出来，并存储于一个叫做 residuals 的数列中，可使用命令 resid(reg) 进行调用。

下面， plot(resid(reg)) 语句是用误差的绘图来确定误差的变化是否比较小范围之内。acf(resid(reg),20) 语句是用来检验误差的自动关联性。 因为我们知道误差之间不应当存在任何关联性。如果误差之间没关联性存在，只有在 lag0 的 ACF 上有数值为 1 的关联性。具体操作和显示如下：

```
par(mfrow=c(2,1))
plot(resid(reg))                    # residuals - reg$resid is same as resid(reg)
acf(resid(reg),20)    #             acf of the resids
```

下面给出了相关误差的两个绘图。 第一个图可以看到误差的变化范围在允许范围之内，并且没大变化。第二张图只有在 lag0 上有数值为 1 的关联性，其他 lag 上的关联性都非常小，即说明误差和误差之间是没关系。这符合我们模型所需要假设的。



2.运用 R 语言研究空气污染

接下来，我们讲解一个延迟线性回归模型，同样要使用到 R语言中 lm() 命令来生成模型。我们以空气污染研究为实例进行讲解。

在实例中，我们要使用空气中的固体污染物含量来预测心血管疾病死亡人数，并且延迟四周，即一个月。心血管疾病死亡人数的数据记为 cmort.dat，空气中固体污染物的数据记为 part.dat 。先将相关数据传到网上，然后用 R语言扫描其连接，进而读取数据。

下面是 R语言读取两个数据所使用的命令语言。

```
mort = ts(scan("http://www.stat.pitt.edu/stoffer/tsa2/data/cmort.dat"),start=1970, frequency=52)
# make these time series objects
Read 508 items
part =
ts(scan("http://www.stat.pitt.edu/stoffer/tsa2/data/part.dat"),start=1970,
frequency=52)
Read 508 items
```

读取数据后，运用 ts.intersect() 命令把 mort 数据、part 数据以及 part 经过延迟四周的数据捆绑在了一起，使三列数据形成行与行之间的正确对应关系，然后进行接线性回归分析。

在这个线性模型中，我们要使用当前空气污染物的数据和四周以前空气污染物数据对当前心血管病死亡的人数进行预测。因为我们使用四个星期以前的数据，因而所建立起来的模型被称为延迟模型，由 R 语言中的 lag（）命令来实现，延迟四个星期则用 -4 表示。

```
ded = ts.intersect(mort,part,part4=lag(part,-4),
dframe=TRUE) #      tie them together in a data frame
```

接下来的语句表示的是如何生成线性模型，使用的也是 lm（）命令。在语句中，mort 数列表示被预测的变量，放在~左边，part 变量和延迟了四周的 part 变量属于用来进行预测的变量，要放在~右边。上面已被捆绑好的数据被称作 ded，要跟在 data = 后面。

```
fit = lm(mort~part+part4, data=ded, na.action=NULL)        #
now the regression will work
```

接下来使用 summary 命令输出所有估计完成的参数。

```
summary(fit)
  Call:
 lm(formula =          mort ~     part +      part4, data        = ded, na.action = NULL)


 Residuals:
      Min                 1Q   Median                3Q      Max
-22.7429         -5.3677  -0.4136   5.2694  37.8539


Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 69.01020    1.37498  50.190  < 2e-16 ***
part        0.15140                      0.02898   5.225 2.56e-07 ***
```

```
part4        0.26297    0.02899                          9.071      < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


Residual standard error: 8.323 on 501 degrees of freedom
Multiple R-Squared: 0.3091,     Adjusted R-squared: 0.3063
F-statistic: 112.1 on 2 and 501 DF,  p-value: < 2.2e-16
```

如上图所示，我们便获得了所研究需要的数据。

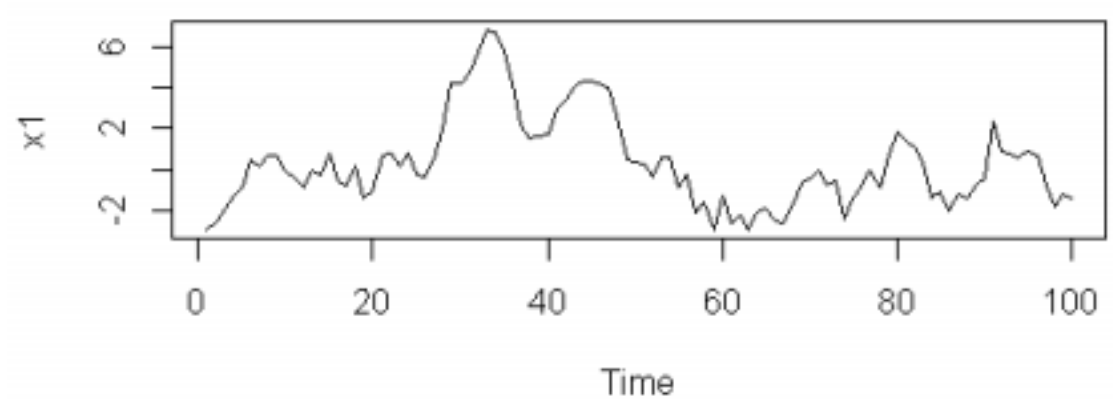
3. 运用 R语言研究自动回归移动平均集成模型

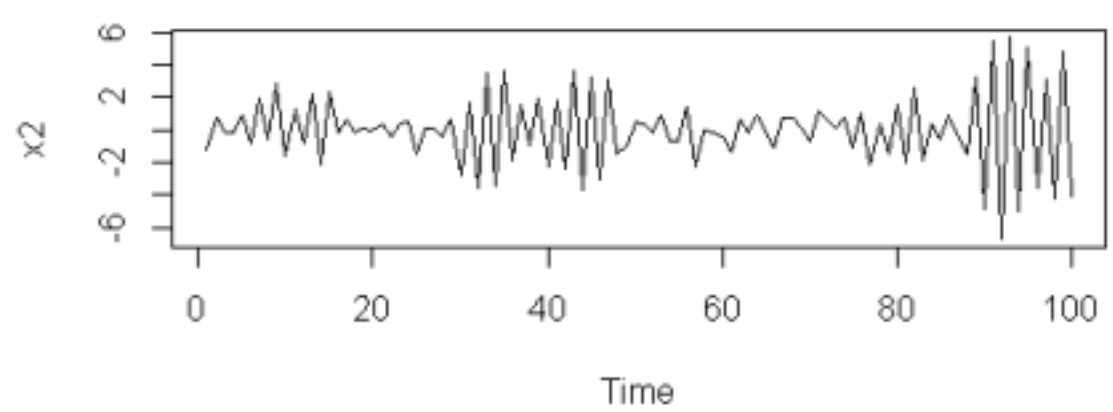
现在来介绍自动回归移动平均集成模型，即 ARIMA模型。AR指自动回归，MA指移动平均，I 指集成。

我们先来生成两个随机的自动回归数列，记为 x1 和 x2。它们都是 ar1 数列，因为生成过程中使用 order=c(1,0,0) 的参数输入。X1 的 AR 参数是 +0.9，x2 的 AR 参数是 -0.9 。

两个语句生成完毕后，我们以将这两个数列绘图出来。两个时间序列非常不同，x1 的参数为 +0.9，它的图像类似于股票价格的变动图形；x2 的参数为 -0.9，它的图像看上去像是一列声波。窗口具体显示如下：

```
x1 = arima.sim(list(order=c(1,0,0), ar=0.9), n=100)
x2 = arima.sim(list(order=c(1,0,0), ar=-0.9), n=100)
par(mfrow=c(2,1))
plot(x1)
plot(x2)
```



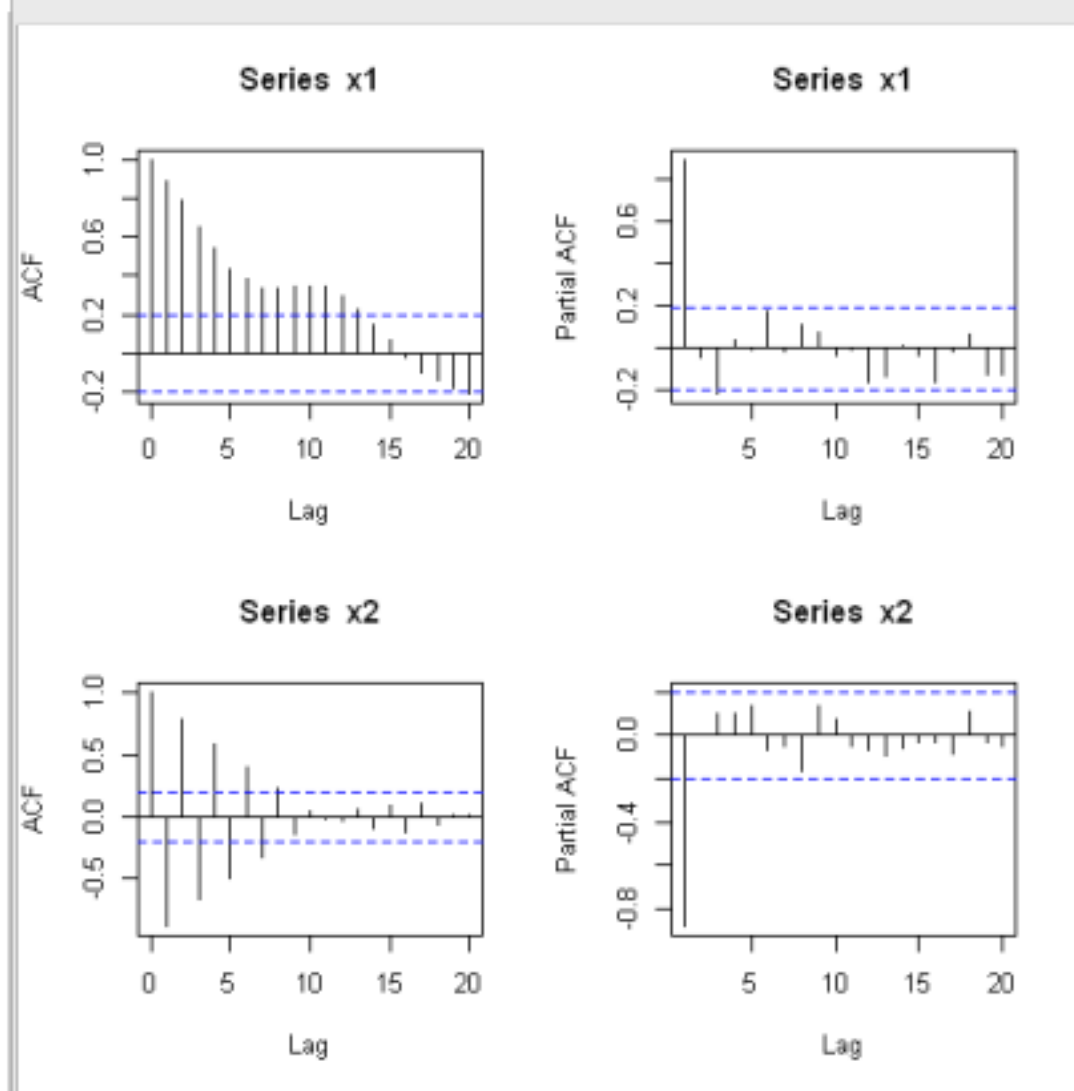


在 R 语言中，ACF指自动关系性，ACF即 Auto-Correlation Function 的简称．比方说，股票价格今天的价格跟昨天的价格有关系，明天的价格会跟今天的或者昨天的价格有关系。它们之间的关系性便用 ACF来衡量。

PACF被称作不完全自动关系性。自动关系性 ACF中存在着线性关系性和非线性关系性。不完全自动关系性就是把线性关系性从自动关系性中消除。如果在线性关系性被去除以后，两个时间点之间的关系性也就是不完全关系性。当 PACF近似于 0，这表明两个时间点之间的关系性是完全由线性关系性所造成的。如果不完全关系性在两个时间点之间不近似于 0，这表明线性模型是不能够表达这两个时间点之间的关系。

我们用下面的语句来检验 x1 和 x2 数列中的 ACF和 PACF。

```
par(mfcol=c(2,2))
acf(x1, 20)
acf(x2, 20)
pacf(x1, 20)
pacf(x2, 20)
```



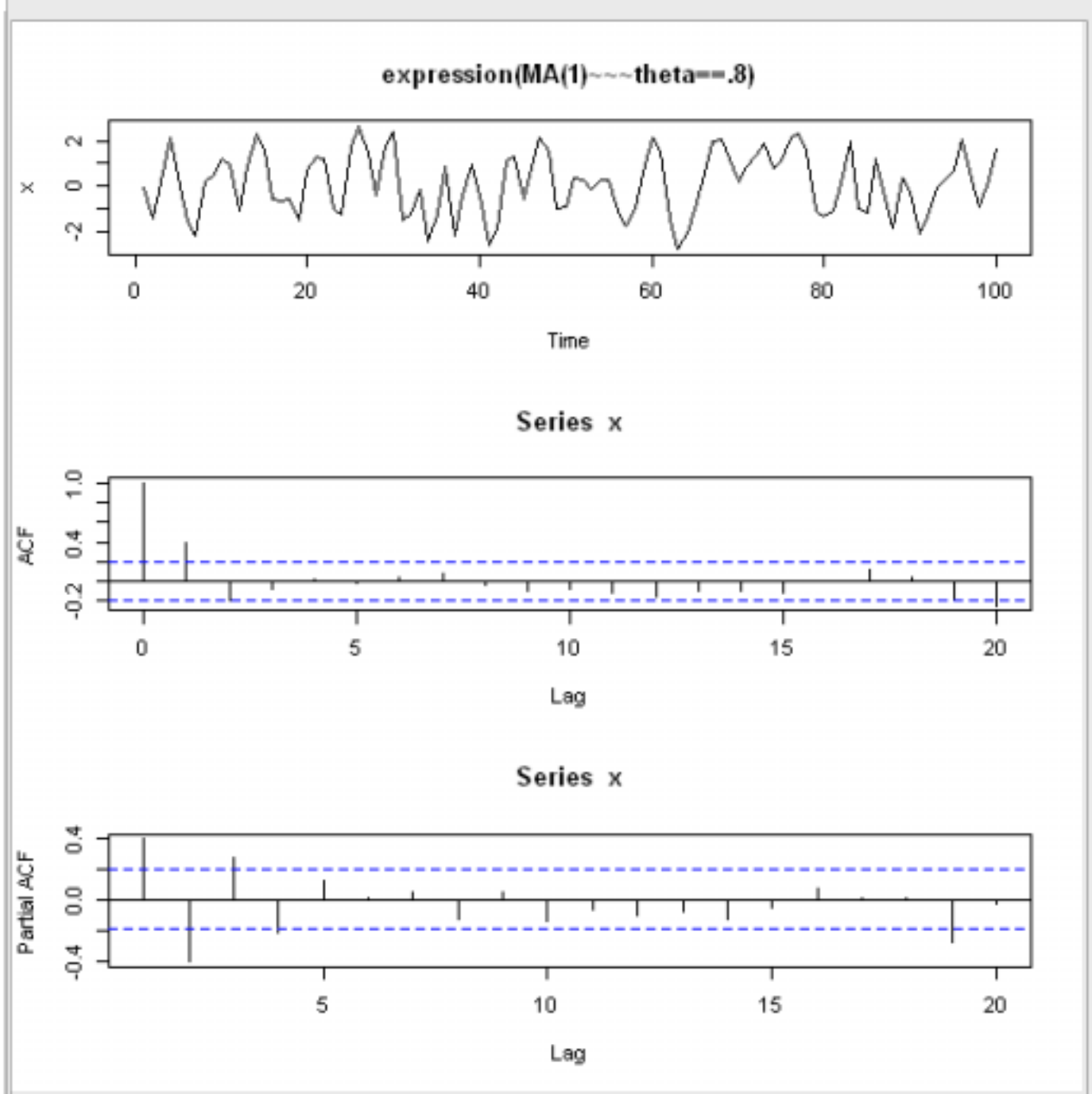
对数列 x1 来讲，ACF从 lag1 到 lag12 都高于蓝色虚线，也就是说两个时间点的距离在 1 到 12 之间它们的自动关系都是正面的。所有线性关系在 x1 数列中被清除了结果是 Partial ACF 的 x1 图表。在 PACF x1图表上可以看到只有 lag0 值为 1，其他的 lag 上的关系值都低于蓝色虚线，近似于 0，也就是说在 x1 数列中存在的自动关系基本上是线性关系。使用线性模型可以符合 x1 数列的要求。

对数列 x2 来讲，两个时间点之间的 ACF关系有负面的也有正面的，但经过去除线性关系以后所有不完全自动关系都接近于 0，也就是说 x2 数列中的自动关系也基本上是线性的。使用线性模型符合 x2 数列要求。

下面，我们来介绍移动平均 MA模型。我们先用 ARIMA.SIM() 命令生成一个移动平均的数据列。不同的是， order=c(0,0,1) ，ma1代表生成的数列， ma是一个参数，设置为 +0.8，数列中有 100 个数据。其余 3 行命令是用来设置输出窗口绘制图表和计算 ACF, PACF数值的。

经过一系列操作后，代码及其结果如下所示：

```
# an MA1
x = arima.sim(list(order=c(0,0,1), ma=.8), n=100)
par(mfcol=c(3,1))
plot(x, main=(expression(MA(1)~~~theta==.8)))
acf(x,20)
pacf(x,20)
```

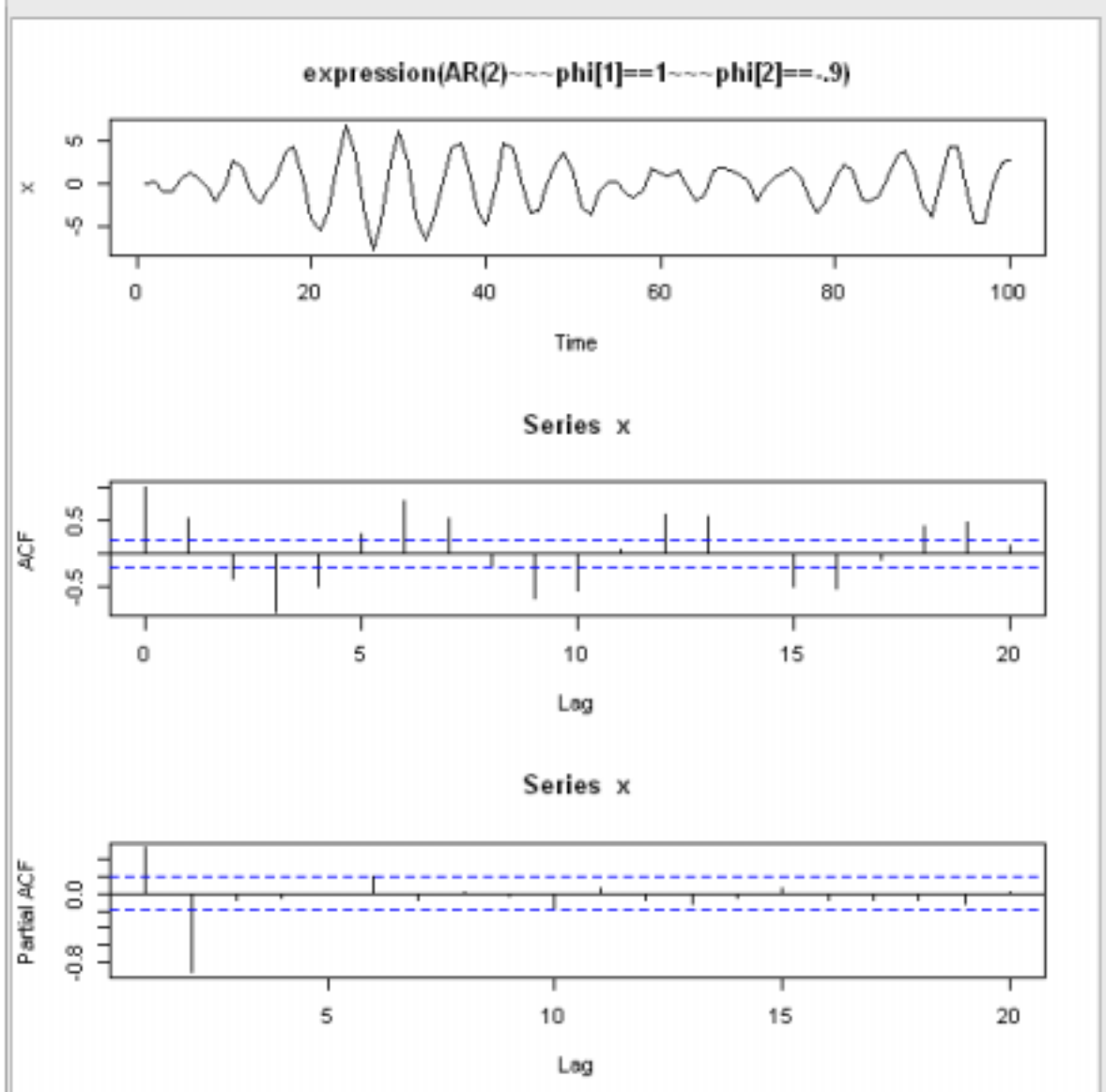


需要注意的是，因数据都是随机生成的，我们在重新运行这些命令时，生成的数据会与以前所生成的数据完全不同， 其结果也不一样。 除非在以上代码的前面使用随机锁定的命令 set.seed(1) ，每次生成的就是相同的数据了。 当然也可以用 set.seed(100) ，和其它数字输入都是可以的。

在上图中，MA1 代表了数列输出的结果。最上面显示的是数列的数目，横向坐标从 1 到 100 代表有 100 个数据。中间的图显示的是 x 数列的自动关系性，在 lag0 上有很强的自动关系性，因为 Lag0 表示没时间延迟，当前数据与它有完全的关系性。所以 lag0 上的关系性什么都不代表。而 Lag1 上存在着正面的关系性，即当前数据与前一个数据的关系性是正面的。其他延迟上的自动关系性都近似于 0，可以不被考虑。

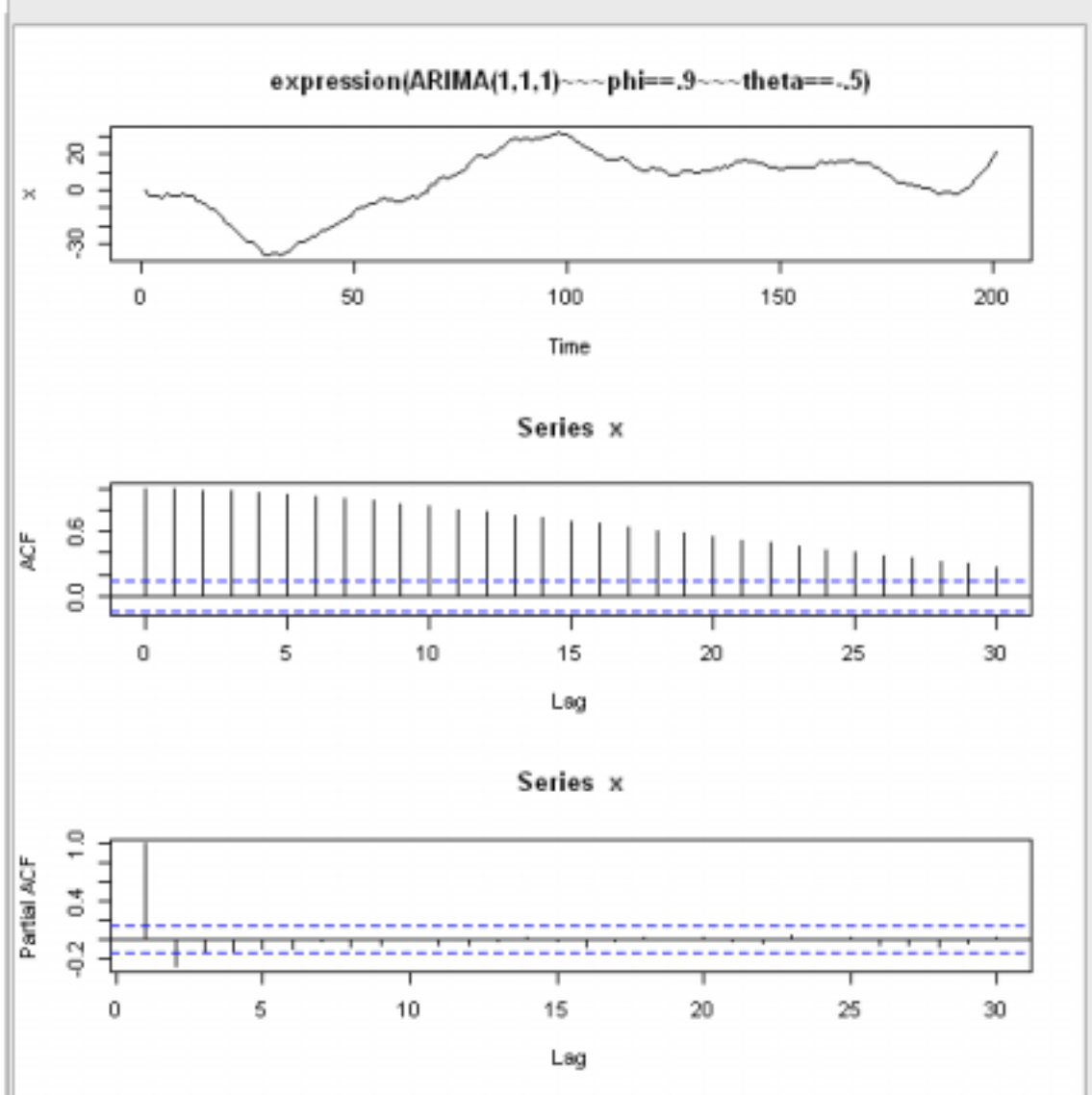
order=c(2,0,0) 代表 AR2 模型。下面，我们来看生成 AR2 数列及分析结果。在这里，需要有两个 AR 参数 1 和-0.9 。输入相关命令后，小窗会有如下显示：

```
# an AR2
x = arima.sim(list(order=c(2,0,0), ar=c(1,-.9)), n=100)
par(mfcol=c(3,1))
plot(x, main=(‘expression(AR(2)~~~phi[1]==1~~~phi[2]==-.9)’))
acf(x, 20)
pacf(x, 20)
```



接下来，我们介绍一下如何生成 ARIMA（1,1,1 ）模型的随机数列。其中的设置 order=c(1,1,1) 说明该数列是 ARIMA 模型，且 AR的参数为 0.9 ，MA的参数为 0.5 ，数列长度为 200 。输入相关命令后，小窗会有如下显示：

```
# an ARIMA(1,1,1)
x = arima.sim(list(order=c(1,1,1), ar=.9, ma=-.5), n=200)
par(mfcol=c(3,1))
plot(x,
main=( " expression(ARIMA(1,1,1)~~~phi==.9~~~theta==-.5) " ))
acf(x, 30) # the process is not stationary, so there is no population [P]ACF ...
pacf(x, 30) # but look at the sample values to see how they differ from the examples above
```



我们再介绍一下自动回归移动平均集成模型的估计分析，即 ARIMA模型。也就是说，我们手头上有一个时间序列数据，如果知道这个数据是被 ARIMA模型生成的，我们希望能估计具体的 ARIMA模型的参数，可以对整个数据生成的过程进行重演和预测。

下面，我们随机生成一些 ARIMA模型的数据，并且假装我们不知道模型的参数，然后做估计练习。

生成 ARIMA数据的命令为 arima.sim(). 自动回归的参数为 0.9，移动平均的参数为 -0.5，估计中要假设这两个参数的设置为未知，并进行估计。计算估计值的命令为 arima() 。其中， x 指随机生成的数据。

在下面的输出中，我们可以看到 AR参数的估计值为 0.8465， ma 的参数估计值为 -0.5021，这两个估计值与前面生成数据时设置的参数非常相似。也就是说，估计得比较精确。

具体操作和显示如下所示：

```
x = arima.sim(list(order=c(1,0,1), ar=.9, ma=-.5), n=100) #
simulate some data
(x.fit = arima(x, order = c(1, 0,                    1))) #      fit the model          and print
the results


Call: arima(x = x, order = c(1, 0, 1))
 Coefficients:
         ar1                  ma1    intercept      <--   NOT the   intercept      -  see  R Issue
1
      0.8465            -0.5021     0.5006
 s.e.  0.0837   0.1356                  0.3150
 sigma^2 estimated as 1.027:  log likelihood = -143.44,                                aic =
294.89
```

下面来介绍衡量参数估计的可信度的方法。 我们对估计的模型进行检测和评估， 需要执行的命令为 tsdiag() 。其中， x.fit 指代估计出来的模型。

这个语句输出的结果如以下几个图表所示。第一个图表代表估计模型误差的绘图。英文叫做 Standardized Residuals， 上面有很多竖线在横向坐标的上下分布。如果这个估计的模型比较可信， 竖线的长度是比较相似的。 如果竖线的长度互相有很大出入或者根本就不同， 估计模型的可信度就非常差。 下面误差绘图中竖线的长度比较相似， 都处在稳定范围之内，即估计的模型没产生不符合要求的误差分布。

再介绍输出的第二张绘图，标题是 ACFof Residuals 。ACF指数据点相互之间的关系，当然在生成这个数据时， 数据点之间互相独立， 并不存在任何关系。 所以在这张图上，只有位于 0 刻度上的竖线最高，其 ACF值为 1。 这个 0 代表数据点与自己相比较，即数据点永远和它自己有关系，这种关系数值为 1。其他横向数轴上的刻度代表一个数据点于其他数据点之间的关系，这些刻度上竖线的长度几乎等于 0，即这个数据点与其他数据点没明显关系。 这张 ACF图代表估计的模型没造成误差之间的任何关系。 这是符合数据生成时每个数据都是独立的这个前提的。由此可见，这 ACF图符合检测要求。

下面来介绍第三张图，也就是 Ljung-Box 指标。这个指标可对每一个时间序列的延迟进行显著性的评估。这张图的横坐标代表时间序列的延迟，纵坐标代表 P-value，即显著性。 如果 P-value 十分小， 就说明在其相对应的延迟点上是显著的。 我们就需要

生成 ARIMA数据的命令为 arima.sim(). 自动回归的参数为 0.9，移动平均的参数为 -0.5，估计中要假设这两个参数的设置为未知，并进行估计。计算估计值的命令为 arima() 。其中， x 指随机生成的数据。

在下面的输出中，我们可以看到 AR参数的估计值为 0.8465， ma 的参数估计值为 -0.5021，这两个估计值与前面生成数据时设置的参数非常相似。也就是说，估计得比较精确。

具体操作和显示如下所示：

```
x = arima.sim(list(order=c(1,0,1), ar=.9, ma=-.5), n=100) #
simulate some data

(x.fit = arima(x, order = c(1, 0,                    1)))   #      fit the model          and print
the results

Call: arima(x = x, order = c(1, 0, 1))

  Coefficients:

         ar1                    ma1    intercept       <--    NOT the   intercept       -  see  R Issue
1

       0.8465            -0.5021     0.5006

  s.e.  0.0837   0.1356                      0.3150

  sigma^2 estimated as 1.027:  log likelihood = -143.44,                                aic =
294.89
```

下面来介绍衡量参数估计的可信度的方法。 我们对估计的模型进行检测和评估， 需要执行的命令为 tsdiag() 。其中， x.fit 指代估计出来的模型。

这个语句输出的结果如以下几个图表所示。第一个图表代表估计模型误差的绘图。英文叫做 Standardized Residuals， 上面有很多竖线在横向坐标的上下分布。如果这个估计的模型比较可信， 竖线的长度是比较相似的。 如果竖线的长度互相有很大出入或者根本就不同， 估计模型的可信度就非常差。 下面误差绘图中竖线的长度比较相似， 都处在稳定范围之内，即估计的模型没产生不符合要求的误差分布。

再介绍输出的第二张绘图，标题是 ACFof Residuals 。ACF指数据点相互之间的关系，当然在生成这个数据时， 数据点之间互相独立， 并不存在任何关系。 所以在这张图上，只有位于 0 刻度上的竖线最高，其 ACF值为 1。 这个 0 代表数据点与自己相比较，即数据点永远和它自己有关系，这种关系数值为 1。其他横向数轴上的刻度代表一个数据点于其他数据点之间的关系，这些刻度上竖线的长度几乎等于 0，即这个数据点与其他数据点没明显关系。 这张 ACF图代表估计的模型没造成误差之间的任何关系。 这是符合数据生成时每个数据都是独立的这个前提的。由此可见，这 ACF图符合检测要求。

下面来介绍第三张图，也就是 Ljung-Box 指标。这个指标可对每一个时间序列的延迟进行显著性的评估。这张图的横坐标代表时间序列的延迟，纵坐标代表 P-value，即显著性。 如果 P-value 十分小， 就说明在其相对应的延迟点上是显著的。 我们就需要

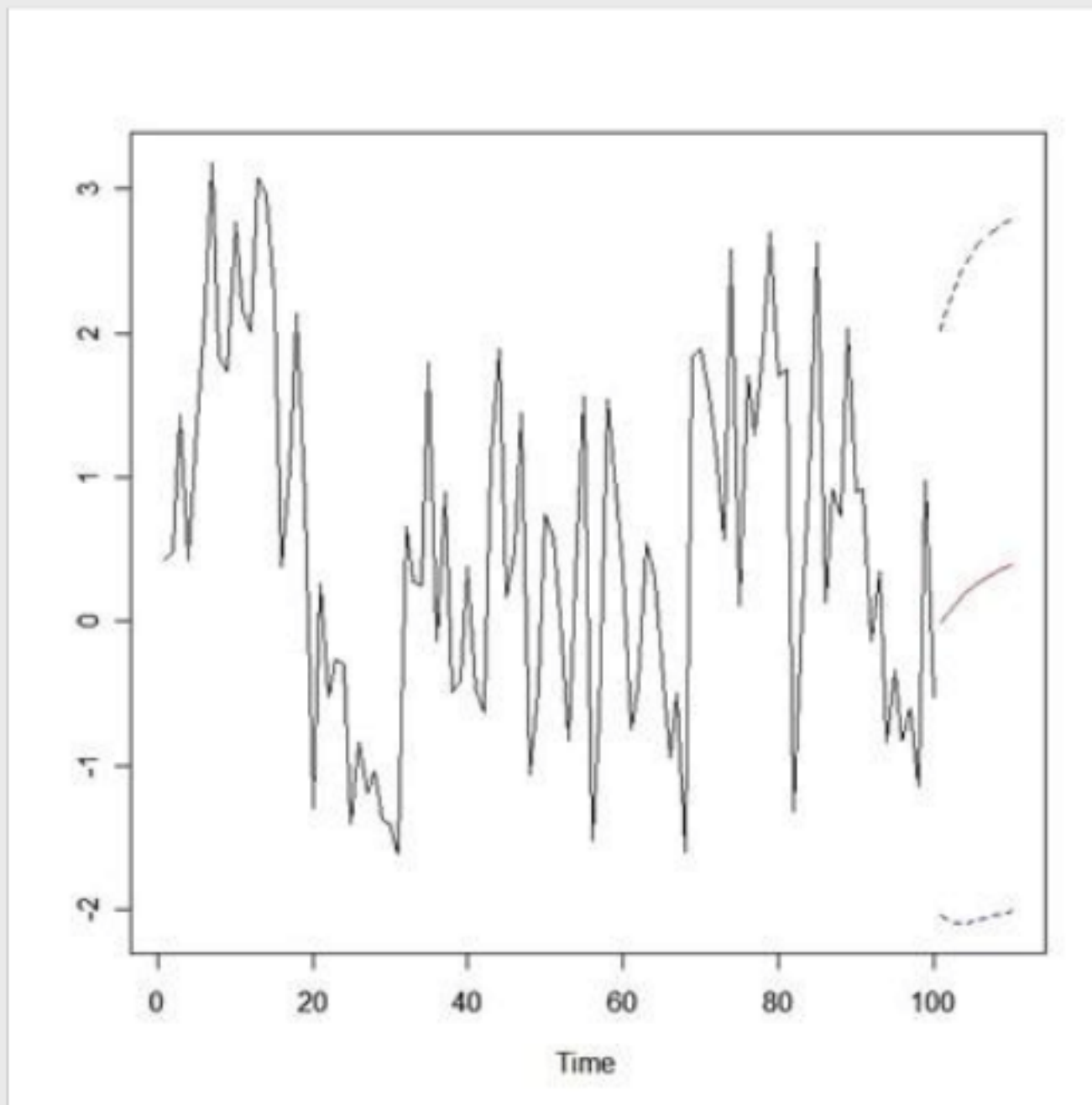
和 n.ahead =10 ，其中 x.fit 代表前面估计出来的模型与参数， n.ahead =10 代表向前预测 10 个数据点。这条语句给我们生成的预测程序项目，被命名为 x.fore.

在下面两条语句中，计算了两列的新数据，分别被称为 U和 L. U 指最高界限， L 指最低界限。 最高限和最低限告诉我们未来数据有很高可能性事发生在两个界限之间，超出或低于这两个界限的几率并不高。

下一步使用 Min 和 Max 条语句，把数据中的最大值和最小值挑选出来， 为接下来的统计绘图设置图标界限。再使用 TS.PLOT()语句．把原有的数据和向前预测 10 个数据点的数据绘在一张图上。最后使用 lines() 语句，把计算的最高界限和最低界限加到绘好的图上。这两条界限的颜色为蓝色，形式也被设置为虚线的形式。

```
x.fore = predict(x.fit, n.ahead=10)
# plot the forecasts
U = x.fore$pred + 2*x.fore$se
L = x.fore$pred - 2*x.fore$se
minx=min(x,L)
maxx=max(x,U)
ts.plot(x,x.fore$pred,col=1:2, ylim=c(minx,maxx))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
```

下面的绘图是由上面的语句所完成的。 图右侧除去原有的数据外， 又加上了三条曲线。最上面和最下面的曲线是虚线， 是计算出的最高界限和最低界限。 中间的曲线为实线，是对未来的 10 个数据点进行的预测。在计算最高、最低限时，我们使用了一个 2 的系数，也就是说，这两个界限之间的几率为 95%。 在未来的 10 个数据点有 95%的几率是出现在上限和下限之间的。 而出现在两个界限之外的几率小于或等于 5%。



4. 运用 R语言研究全球变暖理论

下面，来介绍 R语言在全球温度变暖研究中的应用。 我们将全球温度变暖研究的数据叫 globtemp2.dat. 数据中有 3 列, 第二列指全球每年温度从 1880 年开始到 2004 年。这个数据已经在网上已公布， 我们只需要用下面语句读取就行。 第一条语句读取全球温度数据生成一个叫做 u 的数列矩阵； 第二条语句读取的第二列数据变成时间序列， 命名为 gtemp ；第三条语句是将 gtemp 数据绘成图来看温度变化的具体情况。

操作以后，窗口里会有如下显示：

```
u =read.table("http://www.stat.pitt.edu/stoffer/tsa2/data/globtemp2.dat")  # read the data
gtemp = ts(u[,2], start=1880, freq=1)                    # yearly temp in col 2
plot(gtemp)                                  # graph the series (not shown here)
```

4. 运用 R语言研究全球变暖理论

下面，来介绍 R语言在全球温度变暖研究中的应用。 我们将全球温度变暖研究的数据叫 globtemp2.dat. 数据中有 3 列, 第二列指全球每年温度从 1880 年开始到 2004 年。这个数据已经在网上已公布， 我们只需要用下面语句读取就行。 第一条语句读取全球温度数据生成一个叫做 u 的数列矩阵； 第二条语句读取的第二列数据变成时间序列， 命名为 gtemp ；第三条语句是将 gtemp 数据绘成图来看温度变化的具体情况。

操作以后，窗口里会有如下显示：

```
u =read.table("http://www.stat.pitt.edu/stoffer/tsa2/data/globtemp2.dat")  # read the data
gtemp = ts(u[,2], start=1880, freq=1)  # yearly temp in col 2
plot(gtemp)  # graph the series (not shown here)
```

4. 运用 R语言研究全球变暖理论

下面，来介绍 R语言在全球温度变暖研究中的应用。 我们将全球温度变暖研究的数据叫 globtemp2.dat. 数据中有 3 列, 第二列指全球每年温度从 1880 年开始到 2004 年。这个数据已经在网上已公布， 我们只需要用下面语句读取就行。 第一条语句读取全球温度数据生成一个叫做 u 的数列矩阵； 第二条语句读取的第二列数据变成时间序列， 命名为 gtemp ；第三条语句是将 gtemp 数据绘成图来看温度变化的具体情况。

操作以后，窗口里会有如下显示：

```
u =read.table("http://www.stat.pitt.edu/stoffer/tsa2/data/globtemp2.dat")  # read the data
gtemp = ts(u[,2], start=1880, freq=1)  # yearly temp in col 2
plot(gtemp)  # graph the series (not shown here)
```

4. 运用 R语言研究全球变暖理论

下面，来介绍 R语言在全球温度变暖研究中的应用。 我们将全球温度变暖研究的数据叫 globtemp2.dat. 数据中有 3 列, 第二列指全球每年温度从 1880 年开始到 2004 年。这个数据已经在网上已公布， 我们只需要用下面语句读取就行。 第一条语句读取全球温度数据生成一个叫做 u 的数列矩阵； 第二条语句读取的第二列数据变成时间序列， 命名为 gtemp ；第三条语句是将 gtemp 数据绘成图来看温度变化的具体情况。

操作以后，窗口里会有如下显示：

```
u =read.table("http://www.stat.pitt.edu/stoffer/tsa2/data/globtemp2.dat")  # read the data
gtemp = ts(u[,2], start=1880, freq=1)  # yearly temp in col 2
plot(gtemp)  # graph the series (not shown here)
```

```
Parameter estimate(s):
      Phi1      Phi2
0.3980566 0.4134305
Coefficients:     Value  Std.Error   t-value                                p-value
(Intercept)              3131.5452   857.2141  3.653166                     3e-04
trend          -1.5444     0.4340 -3.558021    4e-04
 part           0.1503     0.0210                      7.162408          0e+00


# resid    analysis-      we assumed   e_t = φ1 e_t-1 + φ2 e_t-2 + w_t where   w_t is   weight.
```

事实上， ARIMA命令也可以完成上述自动关系误差线性模型。使用 ARIMA命令，其中输入的第一个参数是 mort 数列，表示 Mort 是被预测的变量，第二个输入的是 Order=c(2,0,0) 代表 AR2误差模型，xreg=cbind(trend,part) 代表用时间和空气污染物作为进行预测的变量。

下面我们用 ARIMA命令从新估计这个模型，所输出的模型参数可以看出截距为 3132.7085，时间的影响是 -1.5449 ，空气污染物的影响是 0.1503。这些数值与我们上面得到的结果之间略有差异是由计算中四舍五入造成的。

```
 (fit2.gls = arima(mort, order=c(2,0,0), xreg=cbind(trend,
part)))
  Coefficients:
         ar1                ar2     intercept    trend    part
      0.3980             0.4135  3132.7085          -1.5449  0.1503
  s.e.  0.0405  0.0404                 854.6662        0.4328  0.0211
  sigma^2 estimated            as 28.99:        log likelihood = -1576.56,                    aic =
3165.13


Box.test(resid(fit2.gls), 12, type="Ljung")   #                                and so on ...
```

6. 运用 R语言研究估计波动的频率

我们来介绍如何运用 R语言来估计波动的频率。波动是一种非常常见的物理现象。人说话发出声波， 手机通讯通过电波。 我们可以用 R语言对一段被记录下来的波动进行频率的估计。

```
Parameter estimate(s):
    Phi1      Phi2
0.3980566 0.4134305
Coefficients:     Value  Std.Error   t-value                    p-value
(Intercept)    3131.5452   857.2141  3.653166                    3e-04
trend          -1.5444     0.4340 -3.558021    4e-04
part           0.1503     0.0210                7.162408        0e+00
```

resid analysis- we assumed $e_t = \phi_1 e_{t-1} + \phi_2 e_{t-2} + w_t$ where w_t is weight.

事实上，ARIMA命令也可以完成上述自动关系误差线性模型。使用 ARIMA命令，其中输入的第一个参数是 mort 数列，表示 Mort 是被预测的变量，第二个输入的是 Order=c(2,0,0) 代表 AR2误差模型，xreg=cbind(trend,part) 代表用时间和空气污染物作为进行预测的变量。

下面我们用 ARIMA命令从新估计这个模型，所输出的模型参数可以看出截距为 3132.7085，时间的影响是 -1.5449 ，空气污染物的影响是 0.1503。这些数值与我们上面得到的结果之间略有差异是由计算中四舍五入造成的。

```
(fit2.gls = arima(mort, order=c(2,0,0), xreg=cbind(trend,
part)))
Coefficients:
         ar1                ar2     intercept    trend    part
      0.3980            0.4135  3132.7085          -1.5449  0.1503
s.e.  0.0405  0.0404                854.6662        0.4328  0.0211
sigma^2 estimated          as 28.99:        log likelihood = -1576.56,                aic =
3165.13

Box.test(resid(fit2.gls), 12, type="Ljung")   #                          and so on ...
```

6. 运用 R语言研究估计波动的频率

我们来介绍如何运用 R语言来估计波动的频率。波动是一种非常常见的物理现象。人说话发出声波， 手机通讯通过电波。 我们可以用 R语言对一段被记录下来的波动进行频率的估计。

在生成随机波动数据时，我们对于自动回归模型的输入设置为 2，即 AR(2)。假设我们已知模型为自动回归，但不知道其设置为 2，我们也可以使用不完全的信息进行频率估计。但是由于不知道其设置为 2，所以频率也只能是估计出来的。具体输入命令为：

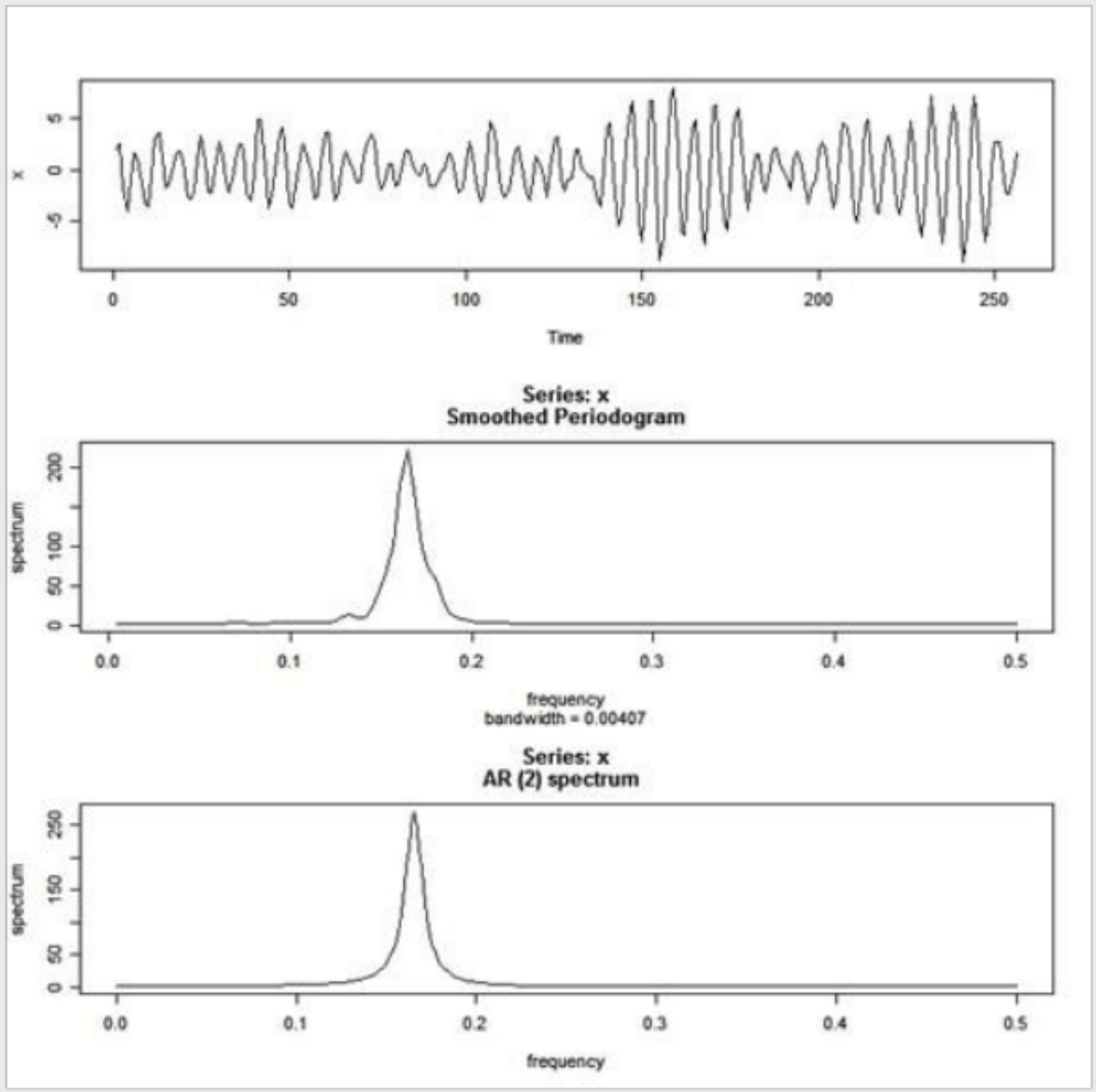
```
spec.ar(x, log="no")          #                    parametric spectral estimate
```

下面是执行上面语句后的输出结果。我们来看窗口中输出的绘成图 3 个图表。最上面图表显示的是随机波动的走势。中间的图表是在假设，用自动回归模型的条件未知下进行的估计。在最顶端峰相对应的水平数轴的值就是我们所估计出来的频率。具体来讲，水平数轴上标有刻度，从 0.00 到 0.5. 山峰最顶端所对应的刻度在 0.16 左右，与前面所计算出的理论值相对应。

最下面的绘图与中间的绘图结论相似。峰值所对应的水平轴刻度也是在 0.16 左右。但这个绘图是在已知自动回归模型，未知模型设置为 2 的条件下估计出来的。所以与中间的估计略有不同，只是更精确。

```
x = arima.sim(list(order=c(2,0,0), ar=c(1,-.9)), n=2^8) #                    some
data
 (u   =  polyroot(c(1,-1,.9)))                          #  x  is   AR(2)   w/complex    roots
   [1] 0.5555556+0.8958064i 0.5555556-0.8958064i
 Arg(u[1])/(2*pi)                                       #  dominant frequency
around .16:
   [1] 0.1616497
par(mfcol=c(3,1))
plot.ts(x)
spec.pgram(x, spans=c(3,3), log="no")  #                    nonparametric spectral
estimate; also see          spectrum()
?spec.pgram                                             #  some help
spec.ar(x, log="no")              #                    parametric spectral
estimate
```

输出的图表：


x
Time
Series: x
Smoothed Periodogram
spectrum
frequency
bandwidth = 0.00407
Series: x
AR (2) spectrum
spectrum
frequency

7.运用 R语言研究厄尔尼诺频率

下面，我们介绍用 R语言研究厄尔尼诺现象的规律。从 1950年到1987年，每一个月都会在中太平洋地区收集空气压力的变化， 用来研究海水表面温度的变化。 所收集的数据被称为南部波动指数，英文叫做 Southern Oscillation Index(SOI) ，因太平洋海水的温度每 3到7年就变暖一次，人称作厄尔尼诺现象。

我们运用 R语言命令直接读取数据，并依次利用下面的语句研究厄尔尼诺现象变化的周期。

第一行 R 语句，是用来读取 SOI 数据的。具体情况如下图：

```
soi = scan("http://www.stat.pitt.edu/stoffer/tsa2 /data/soi.dat")
```

第二行 R语句，是用来读取另外一个数据的，被称作是 rec 。rec 是中部太平洋鱼类数目每月的数据。 rec 的月份与 SOI 数据的月份相对应。具体操作如下：

```
rec = scan("http://www.stat.pitt.edu/stoffer/tsa2/data//recruit.dat")
```

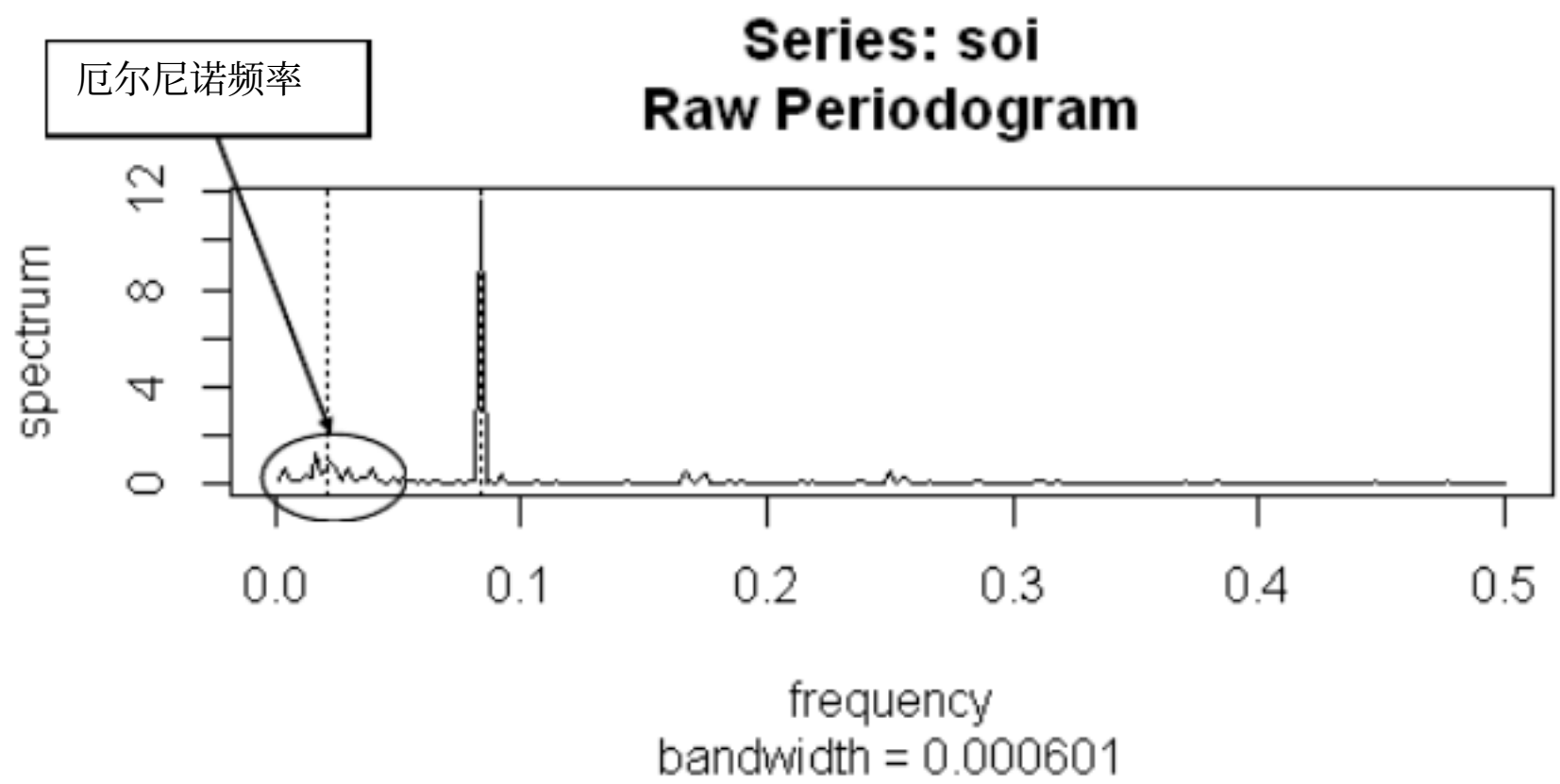
第三行语句，是对 SOI 数据的变化频率进行估计。操作如下：

```
soi.per = spec.pgram(soi, taper=0, log="no")
```

第四条和第五条语句， 是在 1/12 和1/48 频率上加两条数值的虚线。 操作命令如下所示：

```
abline(v=1/12, lty="dotted")
abline(v=1/48, lty="dotted")
```

执行后输出的结果如下。 我们可以看到，SOI 数据有一个很明显的主频率处在 1/12 上，就是说 12 个月为一个周期，很明显这是对应总体气温的年度轮回。然后，我们在下面图上又可以注意到， 存在着一个非常不明显的频率。 竖虚线 1/48 标出了这个频率范围的中心，也就是每四年左右一个循环。前面提到过，厄尔尼诺现象的周期在 3-7 年之间。从下面的图中可以看到，厄尔尼诺频率活动的范围十分地广。



8. 运用 R语言研究太阳黑子周期频率

下面我们介绍运用 R语言对太阳黑子的周期频率进行估计，从而进一步介绍频率估计中的频率混叠，即 aliasing：。 在我们看电影时，我们常会注意到电影中行驶的车子轮胎向后旋转，这是因为电影胶片的拍摄频率为每秒 27 张，而真正现实中汽车轮胎的旋转频率远远高于电影胶片的拍摄频率。 所以在播放时， 汽车轮胎的旋转频率就显得慢了下来，而且方向也被改变了。这种现象叫做频率混叠。

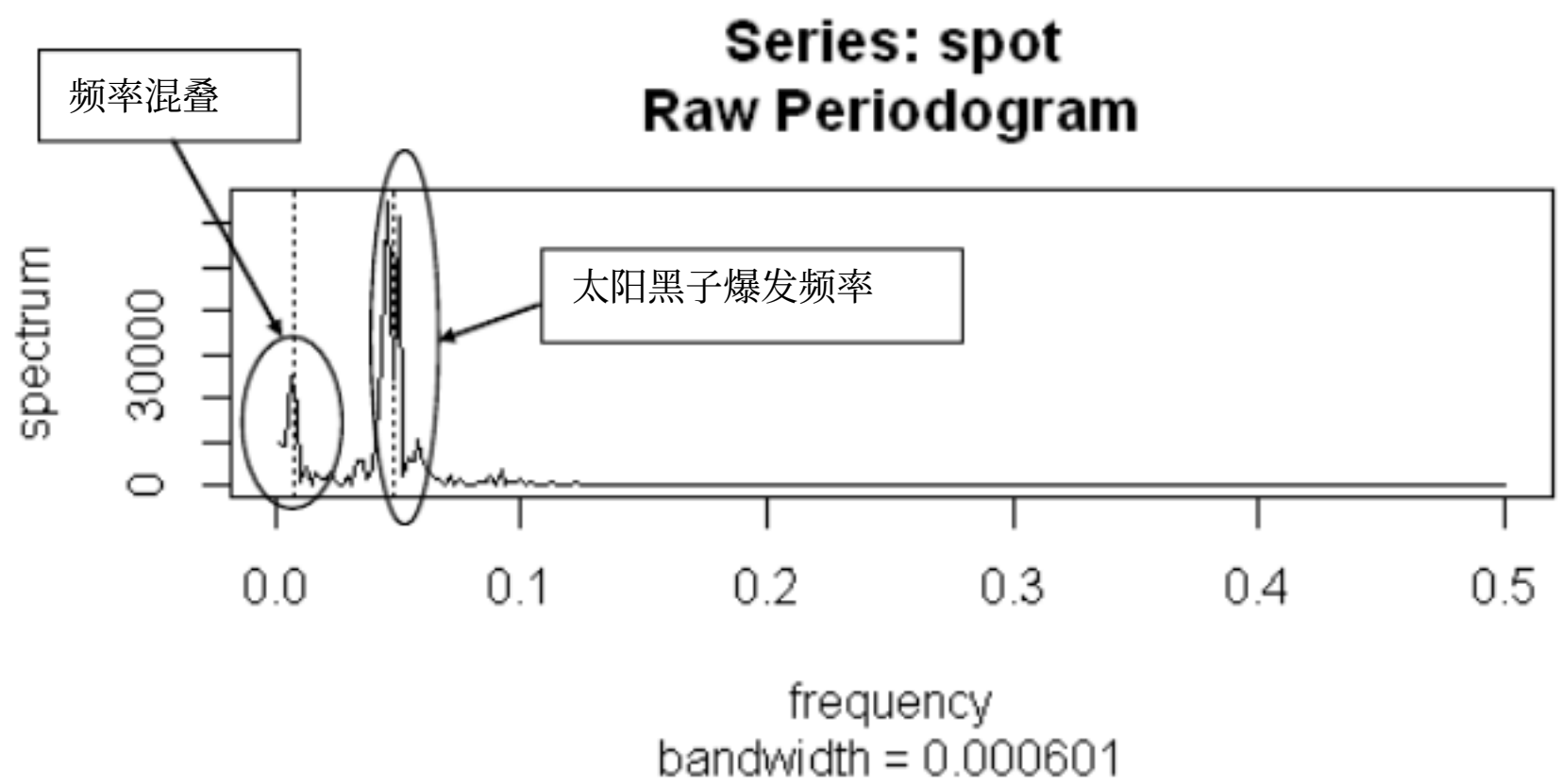
我们这里使用的太阳黑子数据记录是每半年中太阳黑子的数量，即从 1749 年 6 月到 1978 年 12 月，一共是 459 个数据， 也就是说太阳黑子的数量每一年记录两次。 接下来，我们要估计太阳黑子数量的变化周期。 一个太阳黑子从生成到消亡可以有几天或者几个月时间。但总体来讲，太阳黑子数量的变化周期大概在 11 年左右。

用上面我们已经介绍过的频率估计程序， 我们可以生成下面频率图表。 其中主频率在 1/22 上，因为数据是一年收集两次的，与之对应的周期就是 11 年一次。下面图表中还显示了一个比较弱的频率在 1/140 上，这并不是说太阳黑子数量的变化有一个更慢的变化周期， 这是因为太阳黑子生成到消亡有几天到几个月的周期， 明显地快于数据收集的频率。所以，太阳黑子的生成消亡频率就被混叠了， 显示在图表上就变成了一个非常慢的频率。因而在 1/140 上的频率周期指的是单个太阳黑子生成消亡的周期，而不是太阳黑子数量的变化周期。

电脑窗口具体有如下命令：

```
spot = scan("http://www.stat.pitt.edu/stoffer/tsa2/data/sunspots.dat")
spot.per = spec.pgram(spot, taper=0, log="no")
abline(v=1/22, lty="dotted")
```

abline(v=1/140, lty="dotted")



参考： 2006, R.H. Shumway& D.S. Stoffer