Analisi delle Serie Temporali con R

Guido Masarotto
Facoltà di Scienze Statistiche
Università di Padova
guido.masarotto@unipd.it

1 dicembre 2002

Indice

1 Introduzione

Lo scopo di questa scarna nota consiste nel descrivere alcuni degli strumenti disponibili in R (http://www.r-project.org) per l'analisi delle serie temporali. La scelta del materiale è limitata (ed orientata) agli argomenti trattati nel corso di *Analisi delle Serie Temporali* impartito all'interno delle lauree della Facoltà di Scienze Statistiche di Padova.

1.1 Avvertenze

- La nota non contiene nessun richiamo sull'utilizzo, diciamo *base*, di R. Lo studente completamente "digiuno di R" deve perciò utilizzare, prima di questa nota, una delle molte introduzioni disponibili.
- La nota è pensata per essere utilizzata *alla tastiera*, non per essere letta in poltrona.
- Parecchie tra le funzioni considerate hanno altri parametri e/o opzioni. Ovvero, non si eviti di guardare il manuale!

1.2 La biblioteca "ast"

Alcune delle funzioni che utilizzeremo nel seguito sono contenute in una "biblioteca", chiamata ast, che deve essere quindi per prima cosa installata. L'ultima versione della biblioteca può essere trovata all'indirizzo

http://sirio.stat.unipd.it/ts

Il *sorgente* è disponibile nel file `ast_XXX.tar.gz` (XXX indica la versione) e può essere installato, se il sistema è ben configurato, in maniera analoga ad una delle qualsiasi altre biblioteche aggiuntive di R (si veda la FAQ di R). Di norma è sufficiente scaricare il file in una directory e poi dare il comando

```
> R CMD INSTALL /pathto/ast_XXX.tar.gz
```

se si lavora in un sistema Unix o Linux, o

```
> RCmd INSTALL /pathto/ast_XXX.tar.gz
```

se si stà lavorando in una macchina MsWindows.
Gli utenti MsWindows possono comunque trovare nella stessa pagina una versione della biblioteca già precompilata e pronta per essere installata. Il file si chiama

`ast_XXX.zip`. Per installarlo è sufficiente (i) scaricarlo, (ii) lanciare R, (iii) selezionare dal menu "Packages" la voce "Install package from local zip file..." e (iv) nella finestra di dialogo successiva selezionare il file appena scaricato.
L'installazione, ovviamente, va fatta una volta per tutte (almeno fintanchè non si cambia la versione di R). Per poter accedere alle funzioni nella biblioteca è poi però necessario, all'inizio di ogni sessione, caricarle in memoria con il comando

```
> library(ast)
```

Al caricamento, `ast` carica anche le biblioteche
- modreg: metodi "moderni" di regressione
- ts: serie temporali

che dovrebbero far parte di ogni installazione di R. In particolare, molte delle funzioni presentate fanno parte della biblioteca ts.

2 Rappresentazione di una serie temporale

2.1 ts

ts(y, start=1, frequency=1, deltat=1, end=numeric(0))

- y: valori della serie (vettore nel caso di una serie univariata, matrice nel caso di una serie multivariata)
- start: istante iniziale (ad esempio 1967)
- frequency: numero di osservazioni in un unità di tempo
- deltat: la "distanza" nel tempo tra le osservazioni
- end: istante finale

Una serie temporale può essere memorizzata in un vettore o, nel caso sia multivariata, in una matrice. R però dispone di una struttura *ad hoc*, denominata `ts`, per rappresentare in memoria i valori di una serie temporale insieme ad alcune altre caratteristiche (inizio, fine, eventuale periodicità stagionale,...). Per costruire questa struttura è possibile utilizzare la funzione `ts`.
In una applicazione reale, i valori osservati della serie temporale vengono "letti dall'esterno" utilizzando una delle molte funzioni disponibili per farlo (dal disco fisso, da internet, da un data base, ...). Ma per semplicità, come primo esempio, "generiamo" 24 valori che supporremmo, per il momento, costituire la serie osservata

```
> x <- (1:24)/4 + rnorm(24)
> x
 [1]  0.452031655 -0.194178258  0.243101188 -0.007990146
 [5]  0.744130280  2.911109359  2.232786704  1.687583890
 [9]  2.800810573  1.857238229  3.838114776  4.348430501
[13]  3.724094681  2.789332546  2.940917468  2.302895380
[17]  1.972642782  3.206725925  4.212569841  3.495470390
```

```
[21]  3.132792405  6.765122122  5.915017801  4.414375574
> plot(x)
```

Il grafico è mostrato nella figura 2.1. `x` è ovviamente un vettore di 24 numeri non una serie temporale

```
> is.vector(x)
[1] TRUE
> is.ts(x)
[1] FALSE
```

Trasformiamo `x` in una serie storica.

```
> y <- ts(x)
> is.ts(y)
[1] TRUE
> y
Time Series:
Start = 1
End = 24
Frequency = 1
 [1]  0.452031655 -0.194178258  0.243101188 -0.007990146
 [5]  0.744130280  2.911109359  2.232786704  1.687583890
 [9]  2.800810573  1.857238229  3.838114776  4.348430501
[13]  3.724094681  2.789332546  2.940917468  2.302895380
[17]  1.972642782  3.206725925  4.212569841  3.495470390
[21]  3.132792405  6.765122122  5.915017801  4.414375574
> plot(y)
```

La figura 2.2 mostra il grafico. Si osservi come è "cambiata" la maniera in cui R mostra e disegna gli stessi valori.

Esercizio: Verificare che per ottenere da `y` il grafico di prima è sufficiente usare il seguente comando

```
> plot(y,type="p",xlab="Index",ylab="x")
```

Gli elementi della serie temporale possono essere "estratti" come se `y` fosse un vettore

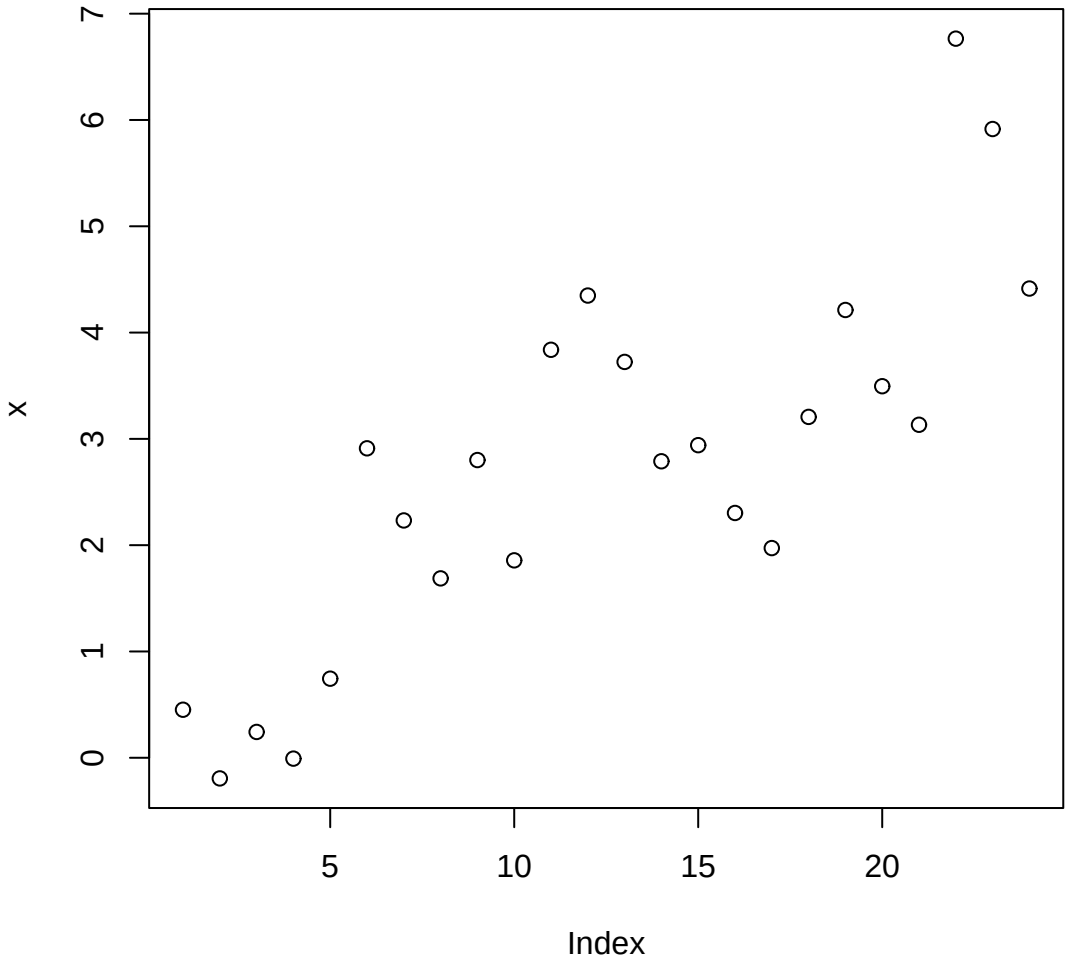

Figura 2.1: Primi dati

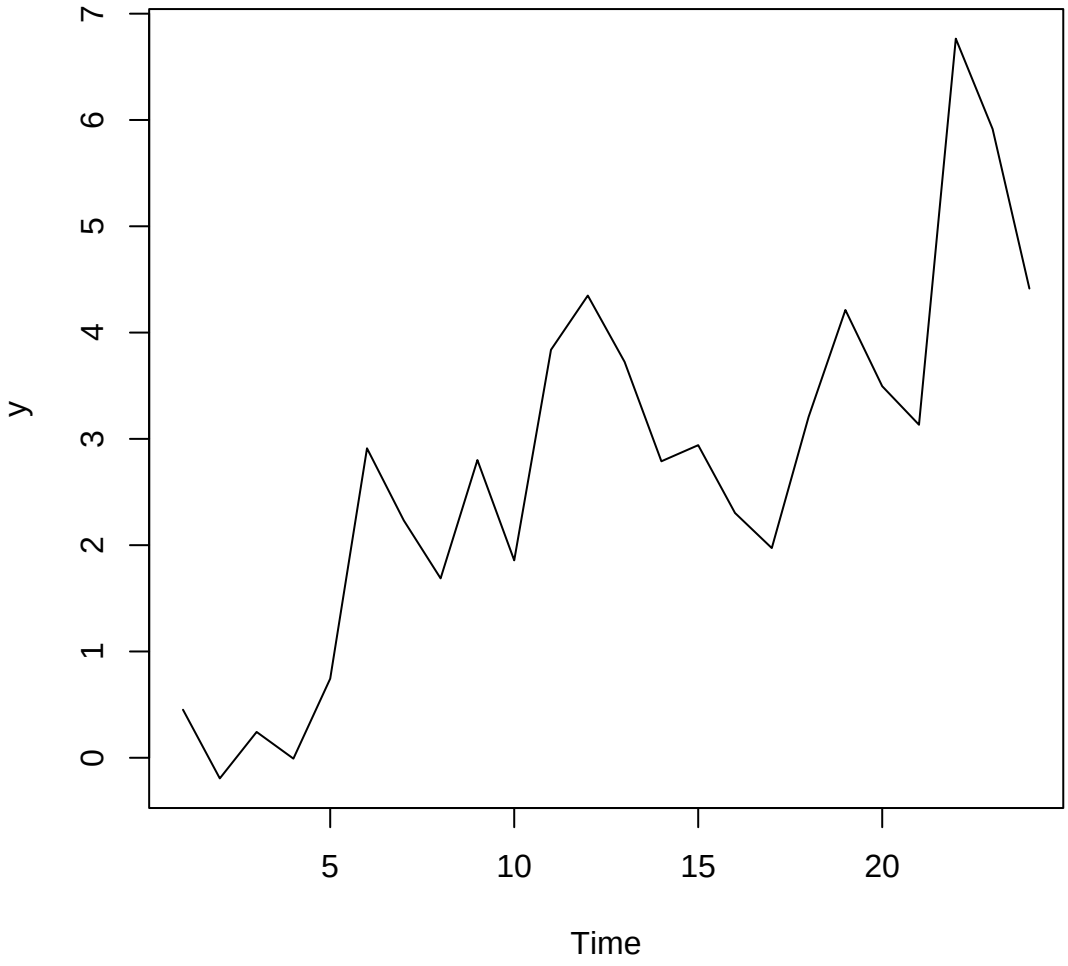

Figura 2.2: Una prima serie storica

```
> y[2]
[1] -0.1941783
> y[15:18]
[1] 2.940917 2.302895 1.972643 3.206726
> y[(1:24)<7]
[1]  0.452031655 -0.194178258  0.243101188 -0.007990146
[5]  0.744130280  2.911109359
> y[-(1:12)]
 [1] 3.724095 2.789333 2.940917 2.302895 1.972643 3.206726
 [7] 4.212570 3.495470 3.132792 6.765122 5.915018 4.414376
```

Si osservi però che i valori "estratti" non sono più una serie temporale.
Utilizzando la stessa "grammatica" possiamo alterare uno o più dei valori osservati. In questo caso, la serie modificata continua ad essere una serie temporale.

```
> y[c(2,10)] <- 0
> y
Time Series:
Start = 1
End = 24
Frequency = 1
 [1]  0.452031655  0.000000000  0.243101188 -0.007990146
 [5]  0.744130280  2.911109359  2.232786704  1.687583890
 [9]  2.800810573  0.000000000  3.838114776  4.348430501
[13]  3.724094681  2.789332546  2.940917468  2.302895380
[17]  1.972642782  3.206725925  4.212569841  3.495470390
[21]  3.132792405  6.765122122  5.915017801  4.414375574
```

Il parametro opzionale `start` serve per informare R dell'istante a cui deve essere riferita la prima osservazione. Ad esempio, supponiamo che i valori contenuti in `x` siano in realtà una serie storica annuale e che la prima osservazione sia da riferire al 1978. Allora, potevamo utilizzare

```
> y <- ts(x, start=1978)
> y
Time Series:
Start = 1978
End = 2001
```

```
Frequency = 1
 [1]  0.452031655 -0.194178258  0.243101188 -0.007990146
 [5]  0.744130280  2.911109359  2.232786704  1.687583890
 [9]  2.800810573  1.857238229  3.838114776  4.348430501
[13]  3.724094681  2.789332546  2.940917468  2.302895380
[17]  1.972642782  3.206725925  4.212569841  3.495470390
[21]  3.132792405  6.765122122  5.915017801  4.414375574
> plot(y)
```

La figura 2.3 mostra il grafico. Si osservi come R abbia calcolato automaticamente l'anno finale e l'asse delle ascisse nel grafico.
`deltat` serve per indicare la distanza nel tempo tra le osservazioni. Ovviamente dipende dall'unità di tempo prescelta. Ad esempio, supponiamo che
- le osservazioni in `x` iniziano nel 1978,
- abbiano cadenza semestrale,
- vogliamo continuare ad utilizzare l'anno come unità di tempo.

Allora, possiamo utilizzare

```
> y <- ts(x, start=1978, deltat=0.5)
> y
Time Series:
Start = c(1978, 1)
End = c(1989, 2)
Frequency = 2
 [1]  0.452031655 -0.194178258  0.243101188 -0.007990146
 [5]  0.744130280  2.911109359  2.232786704  1.687583890
 [9]  2.800810573  1.857238229  3.838114776  4.348430501
[13]  3.724094681  2.789332546  2.940917468  2.302895380
[17]  1.972642782  3.206725925  4.212569841  3.495470390
[21]  3.132792405  6.765122122  5.915017801  4.414375574
```

Si osservino i "campi" `Start, End` e `Frequency` e si notino le differenze con gli esempi precedenti. La lettura in questo caso è che abbiamo 2 osservazioni per unità di tempo, che la prima osservazione è da riferirsi al I semestre del 1978 e l'ultima al II semestre del 1989.
Equivalentemente potevamo utilizzare

```
> y <- ts(x, start=1978, frequency=2)
```

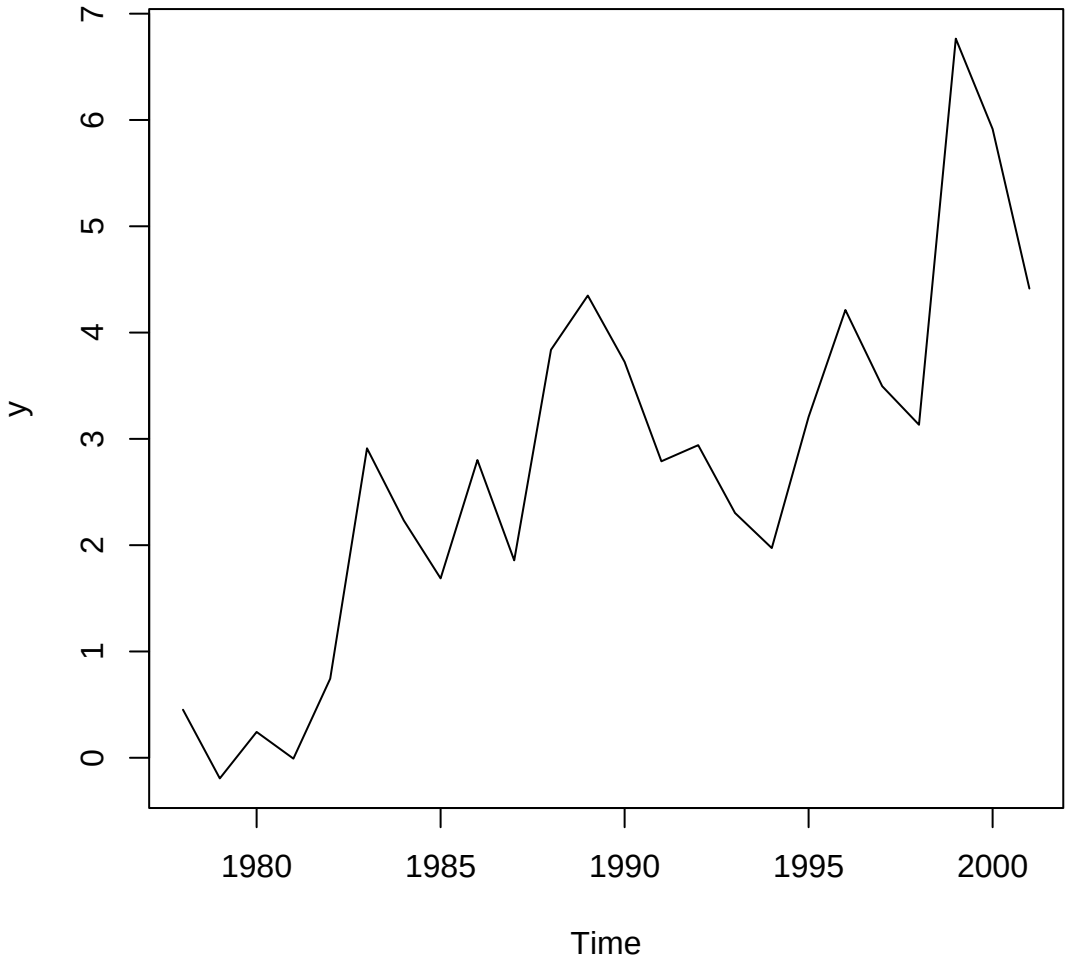

Figura 2.3: La storia inizia. . . nel 1978

```
> y
Time Series:
Start = c(1978, 1)
End = c(1989, 2)
Frequency = 2
 [1]  0.452031655 -0.194178258  0.243101188 -0.007990146
 [5]  0.744130280  2.911109359  2.232786704  1.687583890
 [9]  2.800810573  1.857238229  3.838114776  4.348430501
[13]  3.724094681  2.789332546  2.940917468  2.302895380
[17]  1.972642782  3.206725925  4.212569841  3.495470390
[21]  3.132792405  6.765122122  5.915017801  4.414375574
```

`frequency` rappresenta il reciproco di `deltat` ovvero il numero di osservazioni per unità di tempo.
Come facciamo a dire ad R che la prima osservazione è del II semestre del 1978? Facile. E' sufficiente usare per `start` un vettore di lunghezza due contenente rispettivamente l'anno e il "periodo dentro l'anno".

```
> y <- ts(x, start=c(1978,2), frequency=2)
> y
Time Series:
Start = c(1978, 2)
End = c(1990, 1)
Frequency = 2
 [1]  0.452031655 -0.194178258  0.243101188 -0.007990146
 [5]  0.744130280  2.911109359  2.232786704  1.687583890
 [9]  2.800810573  1.857238229  3.838114776  4.348430501
[13]  3.724094681  2.789332546  2.940917468  2.302895380
[17]  1.972642782  3.206725925  4.212569841  3.495470390
[21]  3.132792405  6.765122122  5.915017801  4.414375574
```

Esercizio: Sempre utilizzando i valori in `x`, come possiamo costruire
- una serie temporale mensile la cui prima osservazione è stata presa nel giugno 1990;
- una serie temporale quadrimestrale che inizia con l'estate del 1721.

Esercizio: Si esplori l'utilizzo del parametro `end`.

2.2 start, end, deltat, frequency

Caratteristiche di una serie temporale
`start(y)`: la data di "inizio" della serie `y`
`end(y)`: la data di "fine" della serie `y`
`deltat(y)`: l'intervallo di tempo tra le osservazioni della serie `y`
`frequency(y)`: il numero di osservazioni per unità di tempo per la serie `y`

Le funzioni nella tabella precedente possono essere utilizzate per "accedere" ad alcune delle caratteristiche della serie temporale.

```
> start(y)
[1] 1978    2
> end(y)
[1] 1990    1
> frequency(y)
[1] 2
> deltat(y)
[1] 0.5
```

Si osservi come il valore ritornato da `start` e `end` sia un vettore di lunghezza due (nel nostro esempio "anno" e "semestre"). Questa forma è sempre utilizzata, anche nei casi in cui ci sia una sola osservazione per unità di tempo

```
> y <- ts( x, start=1756)
> start(y)
[1] 1756    1
> end(y)
[1] 1779    1
```

2.3 time, cycle

Tempo e periodo
`time(y)`: gli “istanti” di tempo della serie `y` `cycle(y)`: le “stagioni” della serie `y`

Un esempio con una serie non “stagionale”...

```
> time(y)
Time Series:
Start = 1756
End = 1779
Frequency = 1
 [1] 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767
[13] 1768 1769 1770 1771 1772 1773 1774 1775 1776 1777 1778 1779
> cycle(y)
Time Series:
Start = 1756
End = 1779
Frequency = 1
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

...e uno con una serie “mensile”...

```
> y <- ts(x, start=c(1996,4), frequency=12)
> y
              Jan           Feb           Mar           Apr
1996                                                0.452031655
1997  1.857238229  3.838114776  4.348430501  3.724094681
1998  6.765122122  5.915017801  4.414375574
              May           Jun           Jul           Aug
1996 -0.194178258  0.243101188 -0.007990146  0.744130280
1997  2.789332546  2.940917468  2.302895380  1.972642782
1998
              Sep           Oct           Nov           Dec
```

```
1996  2.911109359  2.232786704  1.687583890  2.800810573
1997  3.206725925  4.212569841  3.495470390  3.132792405
1998
> time(y)
          Jan      Feb      Mar      Apr      May      Jun
1996                                1996.250 1996.333 1996.417
1997 1997.000 1997.083 1997.167 1997.250 1997.333 1997.417
1998 1998.000 1998.083 1998.167
          Jul      Aug      Sep      Oct      Nov      Dec
1996 1996.500 1996.583 1996.667 1996.750 1996.833 1996.917
1997 1997.500 1997.583 1997.667 1997.750 1997.833 1997.917
1998
> cycle(y)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1996                 4   5   6   7   8   9  10  11  12
1997   1   2   3   4   5   6   7   8   9  10  11  12
1998   1   2   3
```

. . . dovrebbero essere sufficienti. Si osservi, nel secondo esempio, come R conosca "i nomi dei mesi" e quindi adegui automaticamente la maniera in cui la serie temporale viene visualizzata.
Si osservi inoltre che `time` e `cycle` ritornano delle serie temporali.

2.4 window

window(y, start = NULL, end = NULL, frequency = NULL, deltat = NULL, extend = FALSE)

Estrae il "pezzettino" della serie temporale `y` individuato da `start, end, frequency, deltat`, eventualmente estendendo la serie stessa se `extend==TRUE`

`window` permette di estrarre un sottoinsieme delle osservazioni di una serie temporale utilizzando non il loro numero d'ordine (come quando si utilizza l'operatore $[\cdot]$) ma il "tempo". Inoltre, `window` ritorna una serie temporale.
Per mostrare alcuni esempi, "costruiamo" la seguente serie trimestrale

```
> y <- ts(x, start=1990, frequency=4)
> y
             Qtr1         Qtr2         Qtr3         Qtr4
1990  0.452031655 -0.194178258  0.243101188 -0.007990146
1991  0.744130280  2.911109359  2.232786704  1.687583890
1992  2.800810573  1.857238229  3.838114776  4.348430501
1993  3.724094681  2.789332546  2.940917468  2.302895380
1994  1.972642782  3.206725925  4.212569841  3.495470390
1995  3.132792405  6.765122122  5.915017801  4.414375574
```

Incidentalmente, si osservi che, come le serie mensili, anche quelle trimestrali godono di un trattamento speciale[1].
Per "estrarre" le osservazioni fino al II trimestre 1992 possiamo utilizzare

```
> window(y,end=c(1992,2))
             Qtr1         Qtr2         Qtr3         Qtr4
1990  0.452031655 -0.194178258  0.243101188 -0.007990146
1991  0.744130280  2.911109359  2.232786704  1.687583890
1992  2.800810573  1.857238229
```

Viceversa, le osservazioni tra il III trimestre del 1993 e il II del 1995 possono essere ottenute con il comando

```
> window(y, start=c(1993,3), end=c(1995,2))
         Qtr1     Qtr2     Qtr3     Qtr4
1993                     2.940917 2.302895
1994 1.972643 3.206726 4.212570 3.495470
1995 3.132792 6.765122
```

Tutte le osservazioni riferite al secondo semestre sono

```
> window(y, start=c(1990, 2), deltat=1)
Time Series:
Start = 1990.25
End = 1995.25
Frequency = 1
[1] -0.1941783  2.9111094  1.8572382  2.7893325  3.2067259
[6]  6.7651221
```

[1] `Qtr` è l'abbreviazione di *quarter* ovvero trimestre in inglese.

Normalmente `window` non "estende" la serie stessa. Ad esempio, se si richiedono le osservazioni fino al 1996 si ottiene

```
> window(y , end=c(1996,4))
             Qtr1         Qtr2         Qtr3         Qtr4
1990  0.452031655 -0.194178258  0.243101188 -0.007990146
1991  0.744130280  2.911109359  2.232786704  1.687583890
1992  2.800810573  1.857238229  3.838114776  4.348430501
1993  3.724094681  2.789332546  2.940917468  2.302895380
1994  1.972642782  3.206725925  4.212569841  3.495470390
1995  3.132792405  6.765122122  5.915017801  4.414375574
Warning message:
end value not changed in: window.default(x, ...)
```

Si osservi l'avviso finale. Questo comportamento può essere cambiato utilizzando il parametro `extend` come nel seguente esempio

```
> window(y , end=c(1996,4), extend=TRUE)
             Qtr1         Qtr2         Qtr3         Qtr4
1990  0.452031655 -0.194178258  0.243101188 -0.007990146
1991  0.744130280  2.911109359  2.232786704  1.687583890
1992  2.800810573  1.857238229  3.838114776  4.348430501
1993  3.724094681  2.789332546  2.940917468  2.302895380
1994  1.972642782  3.206725925  4.212569841  3.495470390
1995  3.132792405  6.765122122  5.915017801  4.414375574
1996           NA           NA           NA           NA
>
```

Si osservi che la serie è stata "allungata". Al posto dei valori non osservati troviamo dei valori mancanti (`NA`).
Nel caso di una serie temporale non "stagionale" (`frequency(y)==1`), non è necessario utilizzare un vettore bidimensionale per `start` e `end`. Ad esempio,

```
> y <- ts(x, end=1666)
> y
Time Series:
Start = 1643
End = 1666
```

```
Frequency = 1
 [1]  0.452031655 -0.194178258  0.243101188 -0.007990146
 [5]  0.744130280  2.911109359  2.232786704  1.687583890
 [9]  2.800810573  1.857238229  3.838114776  4.348430501
[13]  3.724094681  2.789332546  2.940917468  2.302895380
[17]  1.972642782  3.206725925  4.212569841  3.495470390
[21]  3.132792405  6.765122122  5.915017801  4.414375574
> window(y, start=1650, end=1655)
Time Series:
Start = 1650
End = 1655
Frequency = 1
[1] 1.687584 2.800811 1.857238 3.838115 4.348431 3.724095
```

Esercizio: Si esplori l'utilizzo in `window` del parametro `frequency`.

---

3 Diagrammi di autodispersione, stima della funzione di autocorrelazione, test di Ljung-Box e Box-Pierce

3.1 lag.plot

lag.plot(y, lags = 1, set.lags = 1:lags,...)

- y: serie temporale
- lags: numero di ritardi per cui costruire i grafici
- set.lags: quali ritardi devono essere utilizzati
- ...: altri parametri che controllano la posizione dei grafici (ad esempio su quante righe e colonne) e l'aspetto (vedi il manuale)

La serie delle temperature al Castello di Nottingham il cui grafico è mostrato sul lucido 39 fa parte degli insiemi di dati che "arrivano" con R e che possono essere caricati con il comando `data`. Per ottenere i diagrammi di autodispersione mostrati nei lucidi 42 e 44 è possibile utilizzare i seguenti comandi

```
> data(nottem)
> lag.plot(nottem,set.lags=seq(12,144,by=12))
> lag.plot(nottem,lags=12)
```

Esercizio: Come è possibile ottenere il grafico nel lucido 43?

Esercizio: I dati sulle linci catturate annualmente in Canada mostrati nel grafico sul lucido 4 possono essere caricati con il comando

```
> data(lynx)
```

Usare `lag.plot` per disegnare i logaritmi di y_t verso i logaritmi di y_{t-k} con $k = 1, \ldots, 20$ e commentare il grafico. In particolare, spiegare perchè il grafico ottenuto suggerisce un comportamento ciclico con un periodo di 10 anni.

3.2 acf

acf(y, lag.max = NULL, ...)

- y: serie temporale
- lag.max: sono calcolati i primi `lag.max` coefficienti
- `...`: altri argomenti o per il "sottosistema grafico" o di utilizzo meno comune (si veda il manuale)

L'utilizzo di questa funzione è molto semplice. La funzione calcola le stime della funzione di autocorrelazione e, normalmente, ne produce un grafico. Ad esempio, il grafico nel lucido 40 è stato ottenuto con il comando

```
> acf(nottem,120, ylim=c(-1,1))
```

Si noti l'uso di `ylim` per fissare "l'ampiezza" dell'asse delle ordinate.

Esercizio: Calcolare e commentare il correlogramma della serie delle linci catturate in Canada.

Esercizio: Per simulare un *white noise* (vedi lucidi a pagina 27) gaussiano possiamo utilizzare la funzione `rnorm`. In particolare, le seguenti istruzioni "generano" 300 osservazioni da una serie temporale incorrelata nel tempo, le disegnano e, sullo stesso grafico, aggiungono la stima della funzione di autocorrelazione

```
> old <- par(mfrow=c(2,1))
> y <- ts(rnorm(300))
> plot(y)
> acf(y)
> par(old)
```

Ripeterle un certo numero di volte guardando ogni volta i risultati. Ripetere l'esercizio utilizzando 30 come numerisità. Che cosa si nota di differente?

Esercizio: Le seguenti righe di istruzioni generano una serie temporale con un trend lineare ed una componente erratica incorrelata e gaussiana e ne mostrano il grafico e la funzione di autocorrelazione (stimata). Ripeterle un certo numero di volte eventualmente cambiando i coefficienti della retta che descrive il trend.

```
> old <- par(mfrow=c(2,1))
> a <- 10             #intercetta del trend
> b <- 0.05           #coefficiente angolare del trend
> y <- ts(rnorm(200)) #generiamo la componente erratica
> y <- a+b*time(y)+y  #aggiungiamo il trend lineare
> plot(y)
> acf(y)
> par(old)
```

Esercizio: Le seguenti righe di istruzioni generano una serie temporale caratterizzata dalla presenza di una componente periodica che si ripete ogni "12 mesi". La serie simulata viene disegnata e al grafico viene aggiunta la stima della funzione di autocorrelazione. Ripeterle un certo numero di volte eventualmente cambiando l'ampiezza e la frequenza della "componente stagionale"

```
> old <- par(mfrow=c(2,1))
> a <- 5              #ampiezza della componente stagionale
> f <- 12             #frequenza della componente stagionale
> y <- rnorm(200) #generiamo la componente erratica
> #e sommiamo quella "stagionale"
> y <- ts(a*cos(2*pi*(1:length(y))/f)+y,frequency=f)
> plot(y)
> acf(y,60)
> par(old)
```

3.3 Le linci canadesi

Svolgiamo brevemente i due esercizi precedenti sulle linci canadesi. Innanzitutto "carichiamo" e disegnamiamo i dati

```
> data(lynx)
> plot(lynx)
```

Il grafico è a pagina 4 nei lucidi.
I primi 20 diagrammi di autodispersione della serie originale e del logaritmo della serie stessa possono essere ottenuti con i comandi

```
> lag.plot(lynx,20)
> lag.plot(log(lynx),20)
```

I due grafici sono riportati rispettivamente nelle figure 3.4 e 3.5.
Innazitutto osserviamo come R quando le osservazioni non sono molte connetta i punti con delle linee. Questo può essere evitato se si aggiunge il parametro `do.lines=FALSE` nella chiamata a `lag.plot`.
Le due figure mostrano chiaramente che le relazioni tra "passato" e "presente" sono decisamente più lineari per la serie dei logaritmi. Guardando quindi la figura 3.5 osserviamo come la correlazione sia positiva al primo ritardo poi diventi negativa (in particolare a ritardo 5 e 6) per poi ritornare positiva (con un massimo di "forza" della relazione a ritardo 10) e poi nuovamente negativa... Questo comportamento è legato alla presenza di una componente ciclica evidente nel grafico della serie e che ha un periodo di circa 10 anni. Osservazioni di anni contigui tendono ad essere o ambedue "grandi" o ambedue "piccole" e questo spiega la dipendenza osservata al primo ritardo. Lo stesso vale per osservazioni distanti 10 anni. Mentre l'esatto opposto vale per osservazioni distanti 5/6 anni: se una è nella fase "alta" del ciclo l'altra sarà in quella "bassa" e viceversa.
Per ottenere le stime dei primi 20 coefficienti di autocorrelazione possiamo utilizzare il comando

```
> acf(log(lynx),20)
```

Si osservi che continuamo a lavorare con la serie trasformata. La figura 3.6 mostra il grafico. Il tipo di dipendenza prima descritto diventa ancora più evidente.

3.4 pacf

pacf(y, lag.max = NULL, ...)

- y: serie temporale
- lag.max: sono calcolati i primi `lag.max` coefficienti
- `...`: altri argomenti come per `acf`

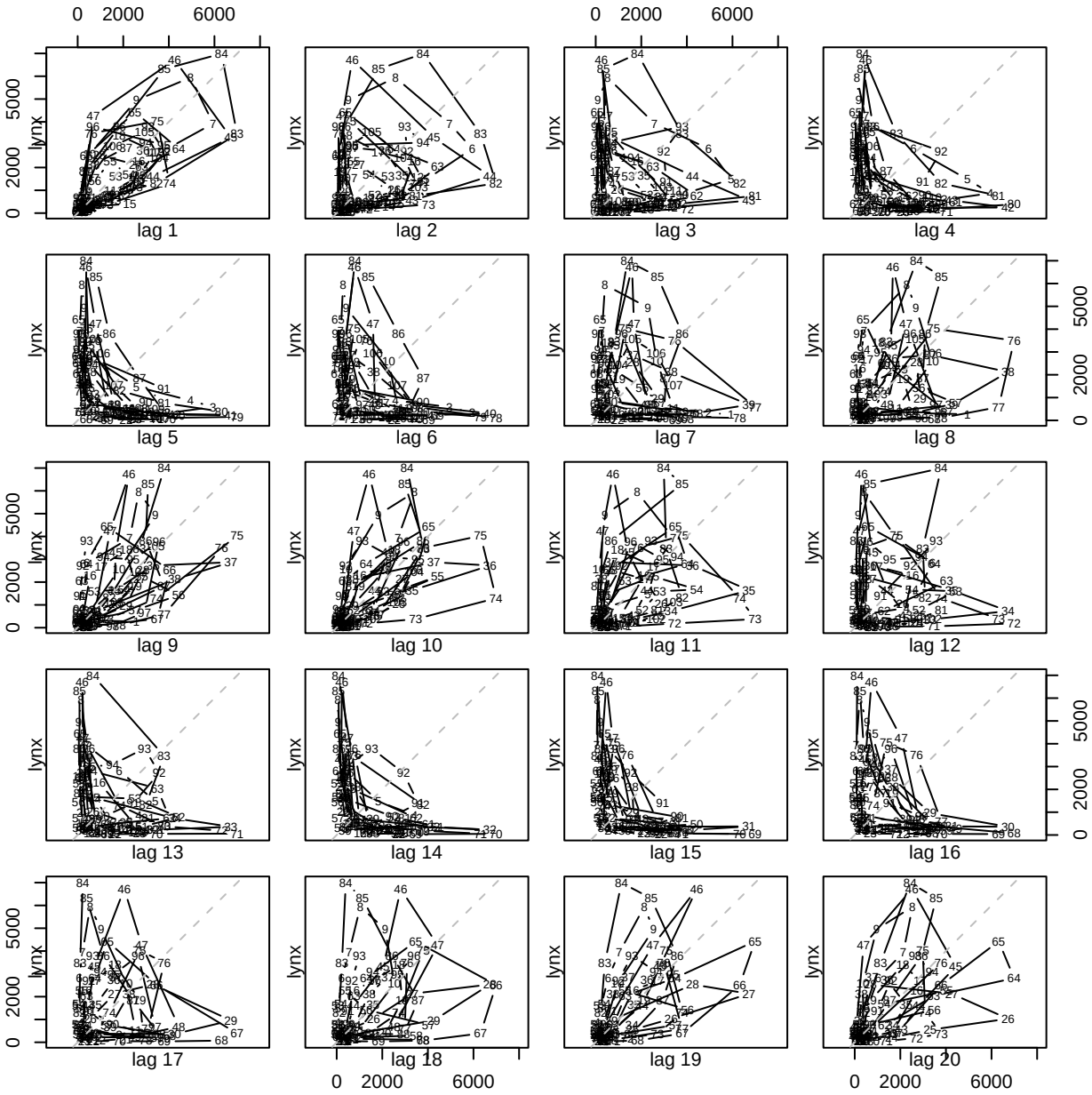

Figura 3.4: Linci canadesi. Diagrammi di autodispersione

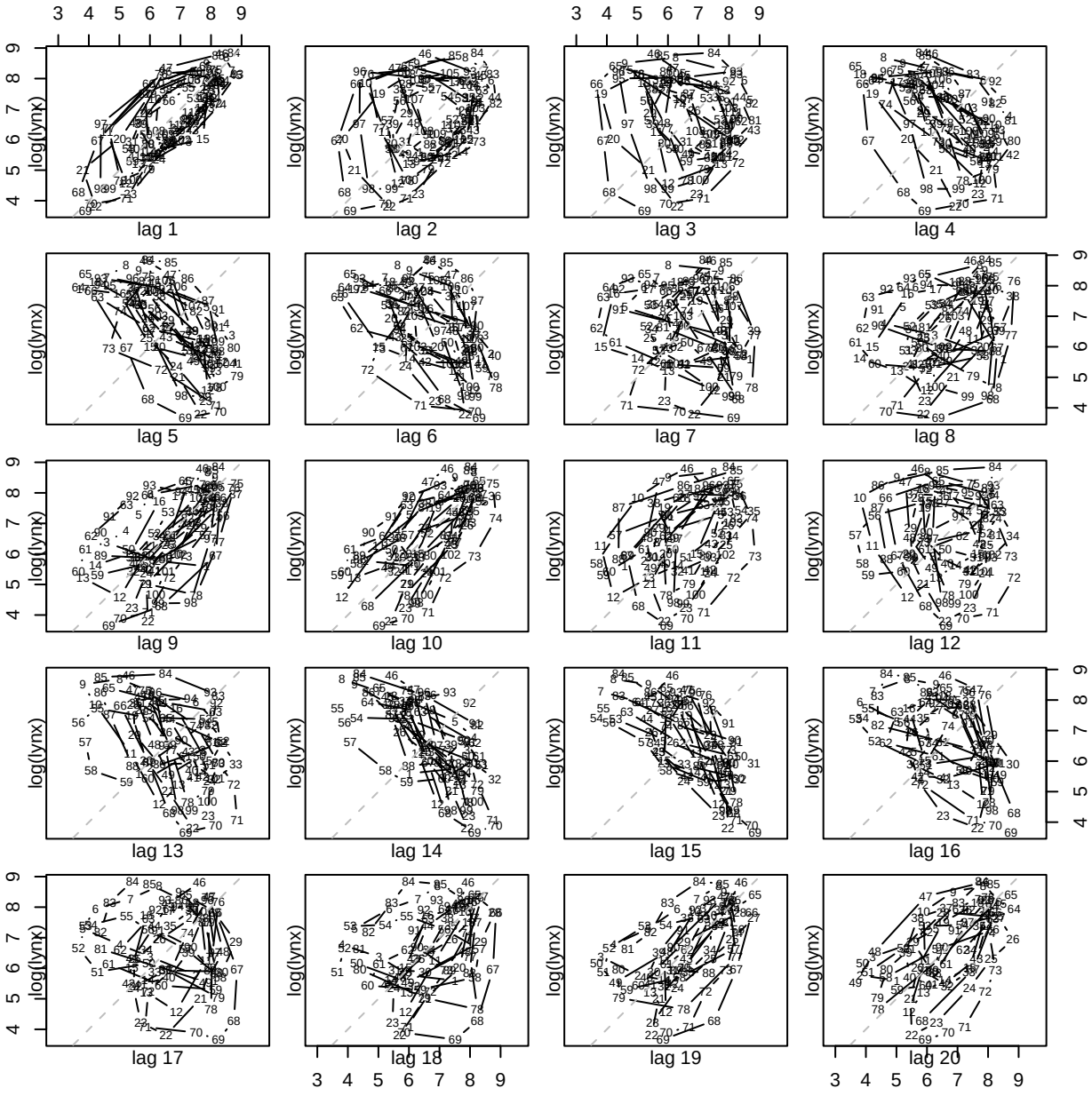

Figura 3.5: Linci canadesi. Diagrammi di autodispersione del logaritmo della serie originale

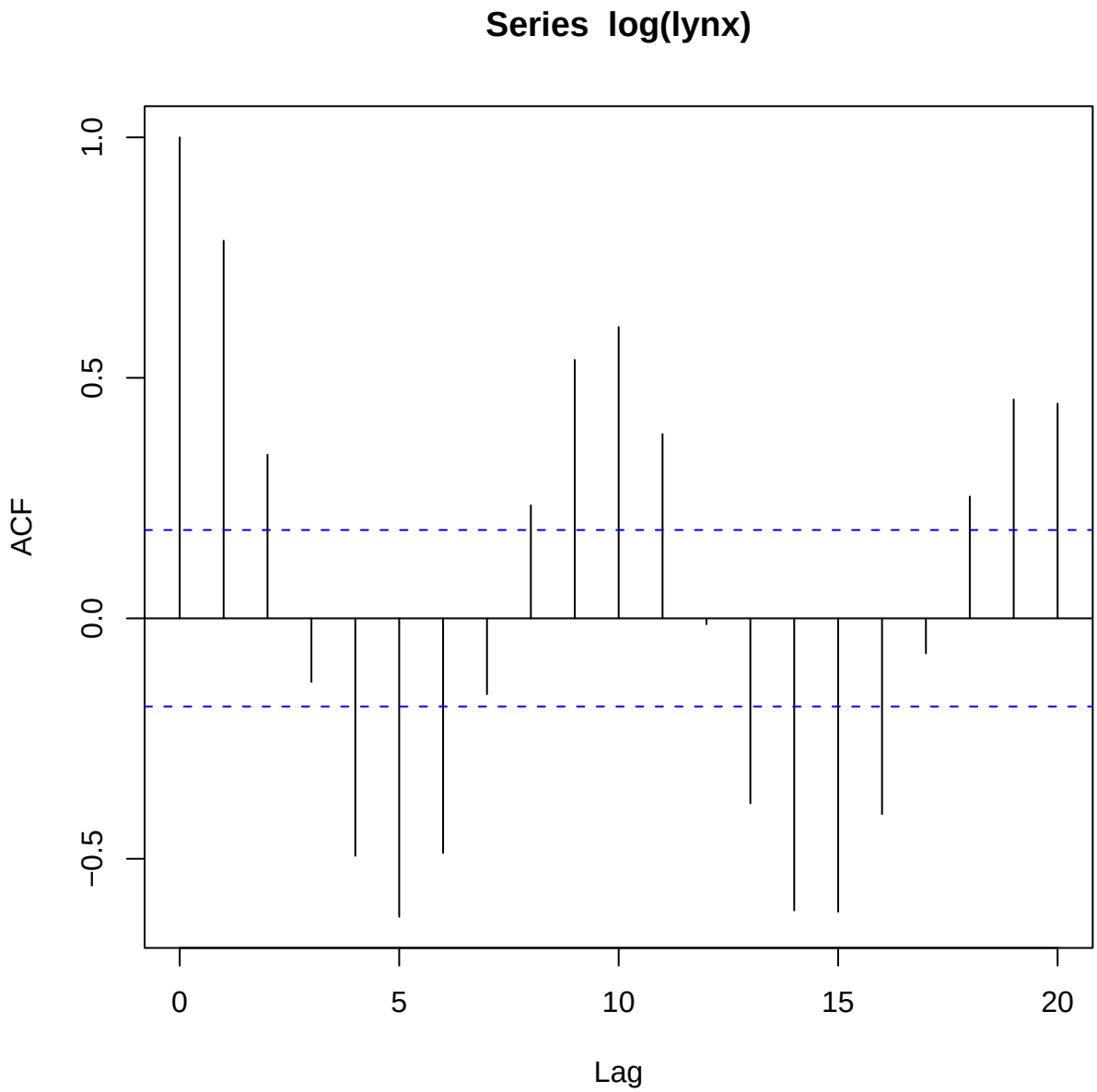

Figura 3.6: Linci canadesi. Correlogramma

Questa funzione, gemella di `acf`, calcola le stime della funzione di autocorrelazione parziale.

3.5 Box.test

Box.test (y, lag = 1, type=c(Box-Pierce, Ljung-Box))

- y: serie temporale
- lag: il test è basato sui primi `lag` coefficienti di autocorrelazione;
- type: tipo di test; si osservi che se questo argomento non viene specificato il test calcolato è quello di Box-Pierce.

Continuamo ad utilizzare la serie delle linci catturate in Canada come esempio.

```
> Box.test(lynx)

        Box-Pierce test

data:  lynx
X-squared = 57.6, df = 1, p-value = 3.209e-14

> Box.test(lynx, type="Ljung-Box")

        Box-Ljung test

data:  lynx
X-squared = 59.1292, df = 1, p-value = 1.477e-14

> Box.test(lynx, lag=10, type="L")

        Box-Ljung test

data:  lynx
X-squared = 215.4452, df = 10, p-value = < 2.2e-16
```

Come si vede quello che R mostra sono, oltre al nome del test e della serie temporale, il valore della statistica (indicata con `X-squared`), il numero dei gradi di libertà (`df`=l'argomento `lag` della funzione) e il livello di significatività osservato (`p-value`). I valori di quest'ultimo sono così piccoli, in tutte le versioni provate, da non lasciar alcun dubbio: la serie delle linci è autocorrelata.
Si osservi, come mostra l'ultimo esempio, che è possibile abbreviare il nome del test.

4 Scomposizione di una serie temporale in componenti elementari

4.1 seaplot

4.2 smoothts

4.3 sfilter

4.4 tsr e amici

4.5 trend, seasonal, remainder, detrend, deseasonal

4.6 residuals, fitted

5 Modelli ARIMA

5.1 ARMAacf, ARMAtoMA

5.2 diff, diffinv

5.3 arima *and friends*

5.4 allArima, bestArima

6 Cenno ad altre funzioni disponibili

6.1 Modelli strutturali

6.2 Lisciamento esponenziale

6.3 Funzioni “specializzate” per i modelli autoregressivi

6.4 Curve non-lineari per il trend

6.5 Modelli GARCH