Chpater 5 Selected Computer Exercises

魏上傑

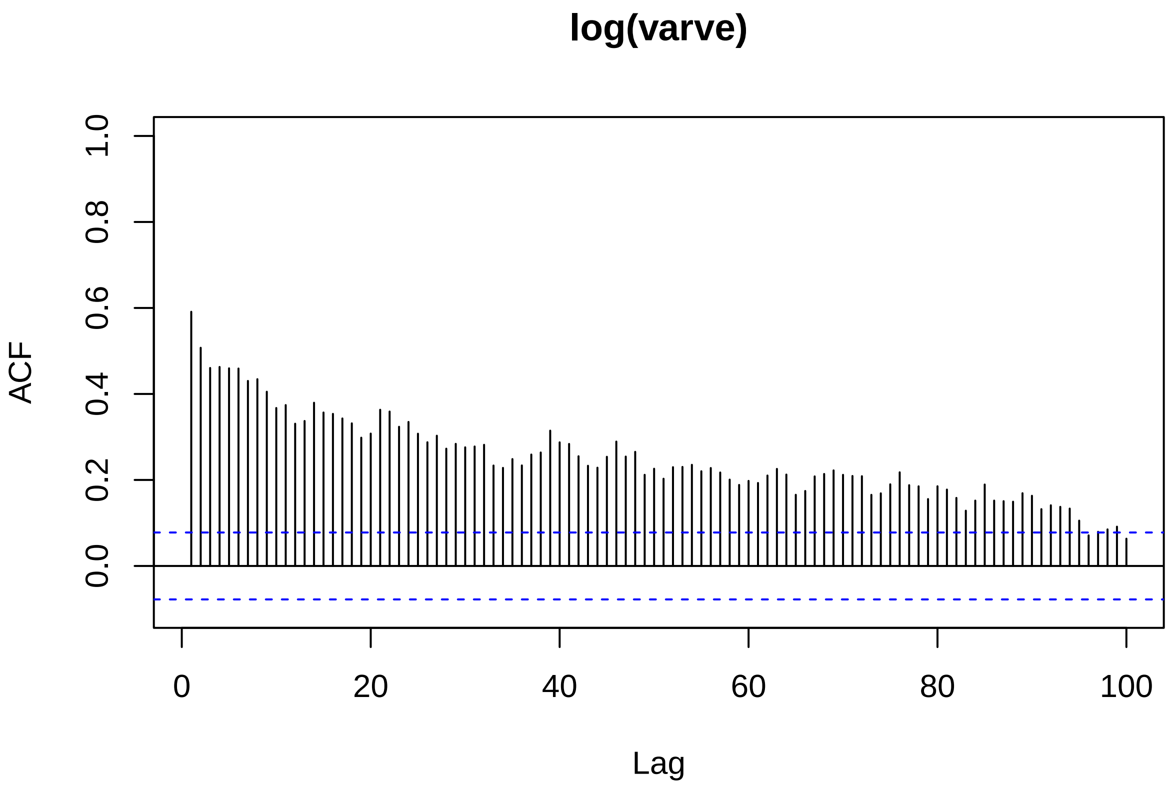
2023-05-04

目錄

1 Section 5.2 Long Memory ARMA and Fractional Differencing

Long memory time series data tend to exhibit sample autocorrelations that are not necessarily large, but persist for a long time.

```
u <- acf(log(varve), 100, plot=FALSE)
plot(u[1:100], ylim=c(-.1,1), main="log(varve)") # get rid of lag 0
```



1.1 Example 5.1

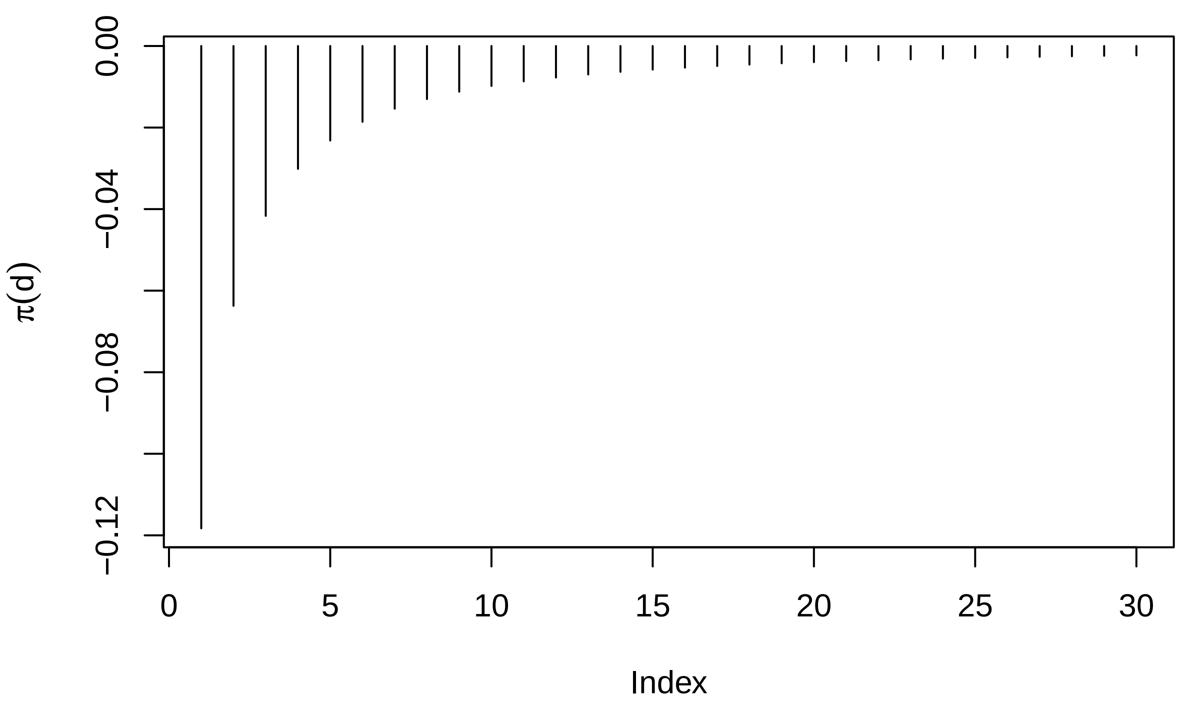
```
library(fracdiff)
lvarve = log(varve)-mean(log(varve)) # mean-adjusted

# M=30, omitting the first 30 points, M=100 is default
varve.fd = fracdiff(lvarve, nar=0, nma=0, M=30)
varve.fd$d # = 0.3841688
```

```
## [1] 0.3841688
```

```
varve.fd$stderror.dpq # = 4.589514e-06 (questionable result!!)
```

```
## [1] 4.589514e-06
```

```
p = rep(1,31)
p[1] <- -varve.fd$d
for (k in 1:30){ p[k+1] = (k-varve.fd$d)*p[k]/(k+1) }
plot(1:30, p[-1], ylab=expression(pi(d)), xlab="Index", type="h")
```




```
p
```

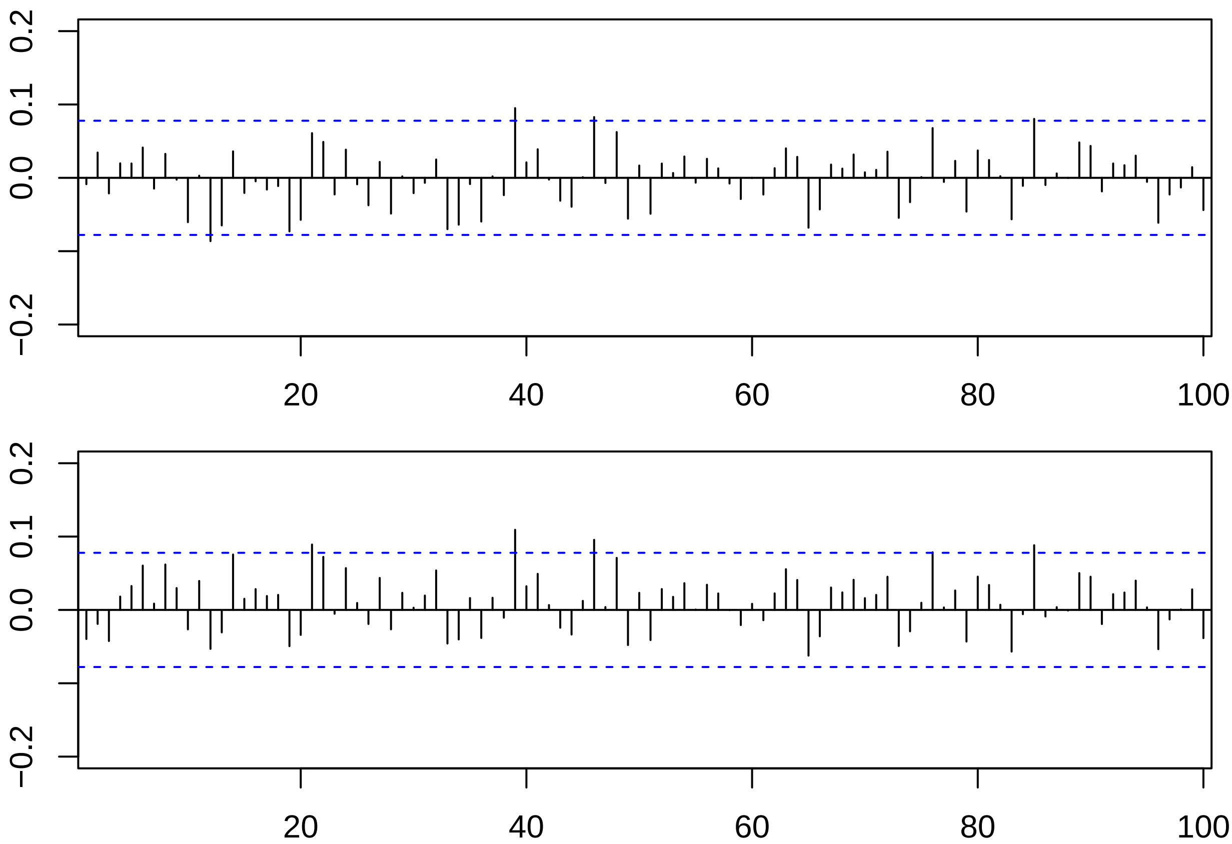
```
##  [1] -0.384168802 -0.118291567 -0.063713068 -0.041665658 -0.030131197
##  [6] -0.023180087 -0.018596493 -0.015378908 -0.013013685 -0.011212371
## [11] -0.009801479 -0.008670904 -0.007747674 -0.006981667 -0.006337414
## [16] -0.005789160 -0.005317797 -0.004908868 -0.004551252 -0.004236267
## [21] -0.003957043 -0.003708078 -0.003484922 -0.003283933 -0.003102112
## [26] -0.002936965 -0.002786400 -0.002648655 -0.002522235 -0.002405861
## [31] -0.002298438
```

We are using $\pi_{j+1}(d) = \frac{(j-d)\pi_j(d)}{j+1}$ for j=0,1,…, with $\pi_0(d) = 1$

Notice that $\pi_{0+1} = \frac{(0-d)\pi_0}{0+1} = -d = -0.384$

The code follows from Chris Builder's notes, the code in Shumway and Stoffer may not be correct.

```
res.fd = diffseries(log(varve), varve.fd$d) # frac diff resids
res.arima = resid(arima(log(varve), order=c(1,1,1))) # arima resids


par(mfrow=c(2,1))
par(mar=c(2,2,1,1))
acf(res.arima, 100, xlim=c(4,97), ylim=c(-.2,.2), main="")
acf(res.fd, 100, xlim=c(4,97), ylim=c(-.2,.2), main="")
```



ACF of residuals from the ARIMA(1, 1, 1) fit to the logged varve series (top) and of the residuals from the long memory model fit, $(1 - B)^d x_t = w_t$, with $d = .384$ (bottom)

The ACFs of the two residual series are roughly comparable with the white noise model.

```
# NOTE: The example in the text uses the package 'fracdiff',
#       which is a dinosaur and gave questionable results -
#       this uses 'arfima' but it didn't make it into the text.

library(arfima)
```

```
## Loading required package: ltsa
```

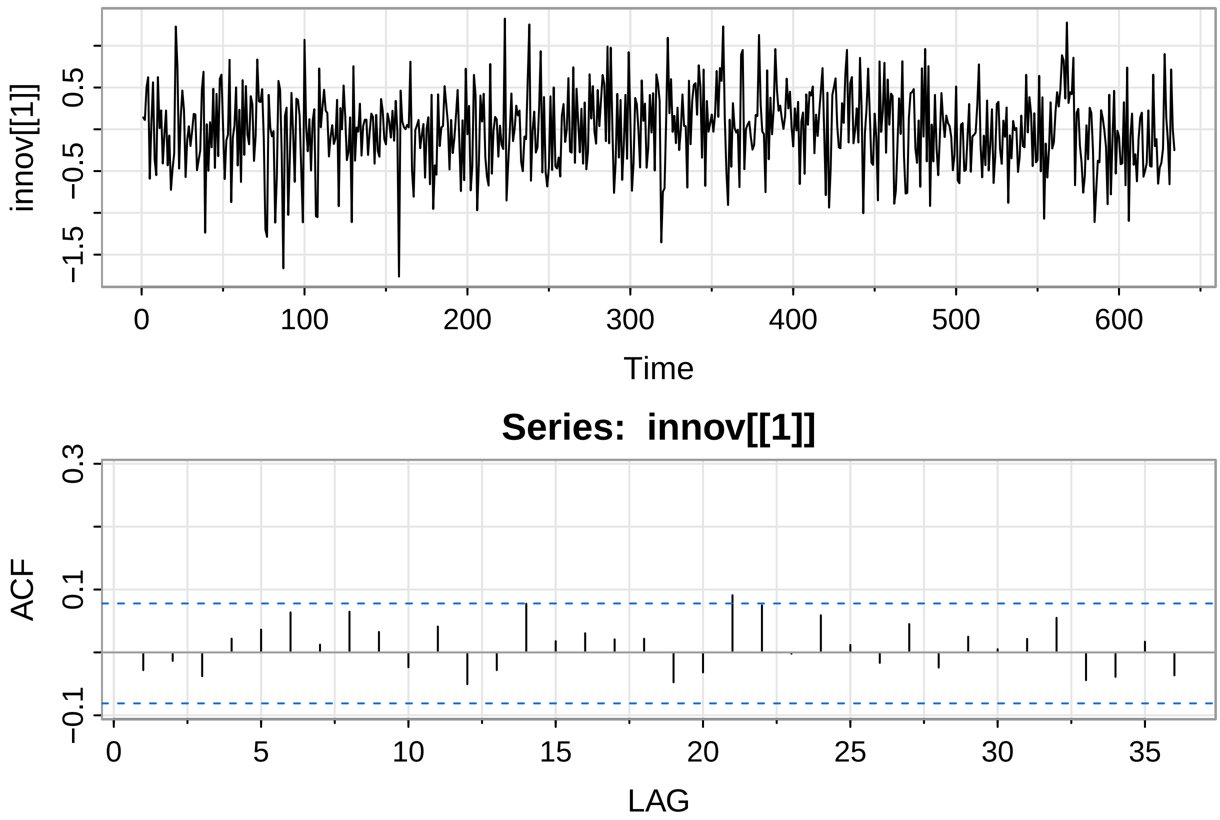
```
## Note that the arfima package has new defaults starting with
```

```
## 1.4-0: type arfimachanges() for a list, as well as some other notes.
## NOTE: some of these are quite important!

##
## Attaching package: 'arfima'

## The following object is masked from 'package:stats':
##
##     BIC
```

```
summary(varve.fd <- arfima(log(varve)))
```

```
## Note: only one starting point.  Only one mode can be found -- this is now the default b
## Beginning the fits with 1 starting values.

##
## Call:
##
## arfima(z = log(varve))
##
##
## Mode 1 Coefficients:
##               Estimate Std. Error Th. Std. Err. z-value   Pr(>|z|)
## d.f          0.3727893  0.0273459     0.0309661 13.6324 < 2.22e-16 ***
## Fitted mean 3.0814142  0.2646507            NA 11.6433 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## sigma^2 estimated as 0.230081; Log-likelihood = 466.028; AIC = -926.056; BIC = -912.699
##
## Numerical Correlations of Coefficients:
##              d.f   Fitted mean
## d.f           1.00 -0.01
## Fitted mean -0.01  1.00
##
## Theoretical Correlations of Coefficients:
##     d.f
## d.f 1.00
##
## Expected Fisher Information Matrix of Coefficients:
##     d.f
## d.f 1.64
```

```
# d.hat = 0.3728, se(d,hat) = 0.0273

# residual stuff
innov = resid(varve.fd)
par(mfrow=2:1)
tsplot(innov[[1]])
acf1(innov[[1]])
```




```
##  [1] -0.03 -0.01 -0.04  0.02  0.04  0.06  0.01  0.06  0.03 -0.02  0.04 -0.05
## [13] -0.03  0.08  0.02  0.03  0.02  0.02 -0.05 -0.03  0.09  0.07  0.00  0.06
## [25]  0.01 -0.02  0.05 -0.02  0.02  0.01  0.02  0.06 -0.04 -0.04  0.02 -0.04
```

2 Section 5.3 Unit Root Testing

$$x_t = \phi x_{t-1} + w_t \tag{1}$$

A unit root test provides a way to test whether the model above is a random walk (the null case) as opposed to a causal process (the alternative). That is, it provides a procedure for testing

$$H_0 : \phi = 1$$
$$H_1 : |\phi| < 1$$

We note that the DF test was established by noting that if $x_t = \varphi x_{t-1} + w_t$, then $\nabla x_t = (\varphi - 1) x_{t-1} + w_t = \gamma x_{t-1} + w_t$, and one could test $H_0 : \gamma = 0$ by regressing ∇x_t on x_{t-1}.

The ADF test was extended to accomodate AR(p) models.

The choice of p is crucial, and we will discuss some suggestions in the example. For ARMA(p, q) models, the ADF test can be used by assuming p is large enough to capture the essential correlation structure; another alternative is the Phillips-Perron (PP) test, which differs from the ADF tests mainly in how they deal with serial correlation and heteroskedasticity in the errors.

Note that in each case, the general regression equation incorporates a constant and a linear trend.

In the ADF test, the default number of AR components included in the model, say k, corresponds to the suggested upper bound on the rate at which the number of lags, k, should be made to grow with the sample size for the general ARMA(p, q) setup.

For the PP test, there is also the default value of k.

2.1 Example 5.3

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
adf.test(log(varve), k=0) # DF test
```

```
## Warning in adf.test(log(varve), k = 0): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  log(varve)
## Dickey-Fuller = -12.857, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(log(varve)) # ADF test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  log(varve)
## Dickey-Fuller = -3.5166, Lag order = 8, p-value = 0.04071
## alternative hypothesis: stationary
```

```
pp.test(log(varve)) # PP test
```

```
## Warning in pp.test(log(varve)): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  log(varve)
## Dickey-Fuller Z(alpha) = -304.54, Truncation lag parameter = 6, p-value
## = 0.01
## alternative hypothesis: stationary
```

In each test, we reject the null hypothesis that the logged varve series has a unit root. The conclusion of these tests supports the conclusion of the previous section that the logged varve series is long memory rather than integrated.

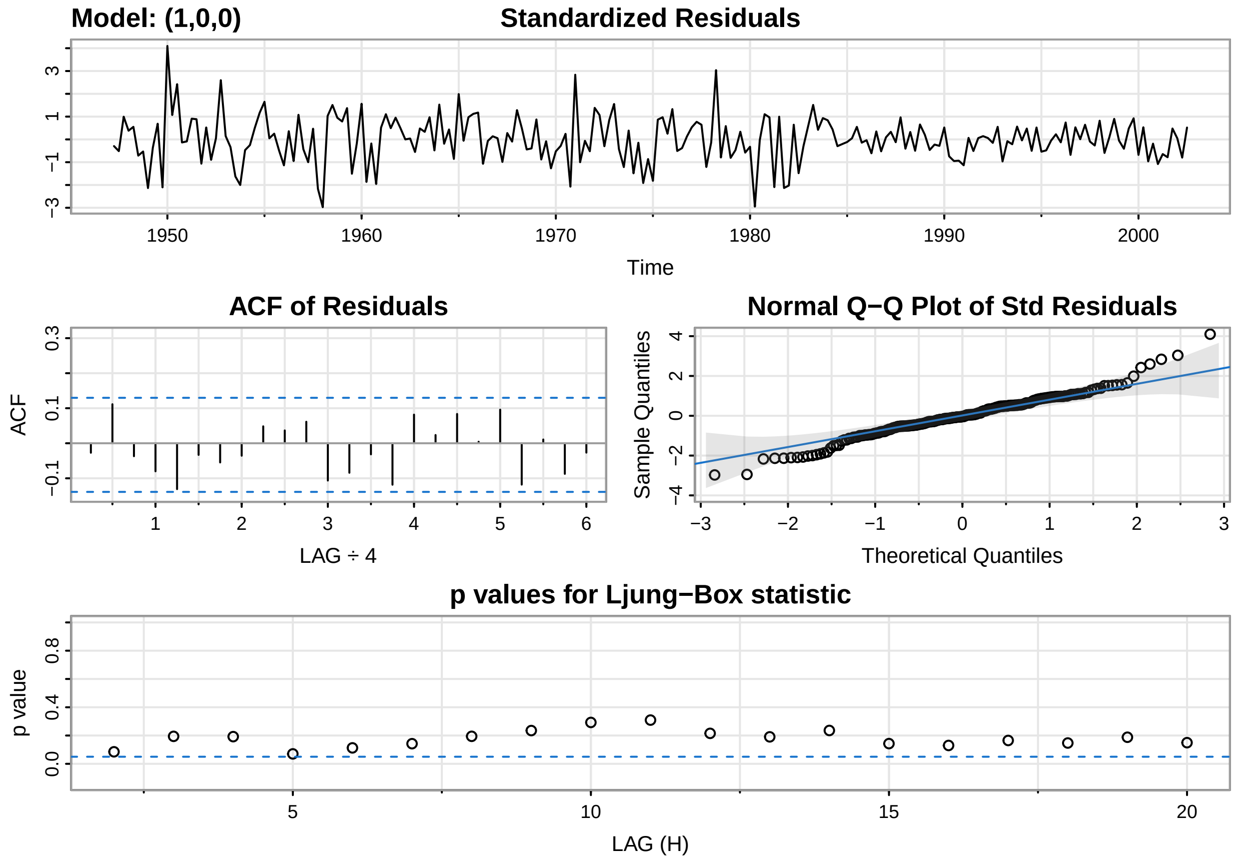
3 Section 5.4 GARCH Models

3.1 Example 5.4

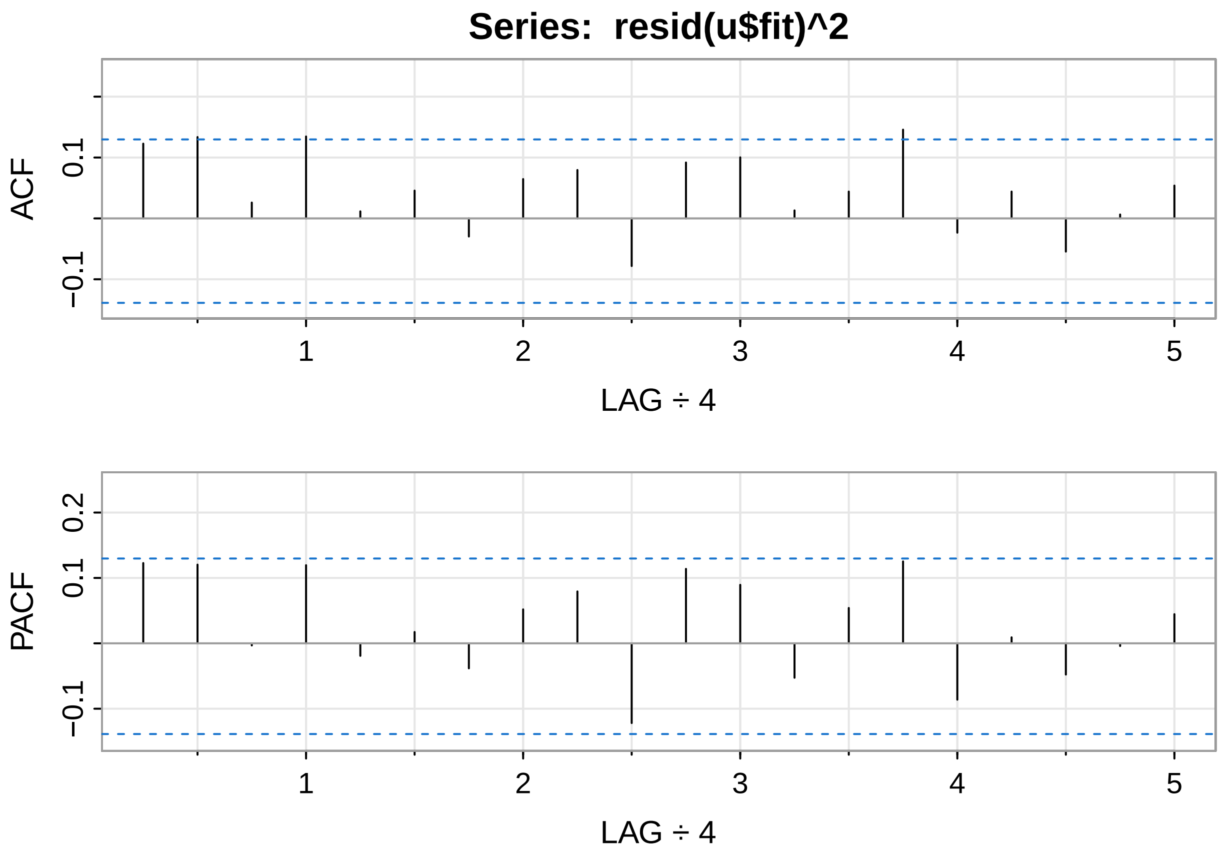
```
gnpgr = diff(log(gnp)) # get the growth rate
u     = sarima(gnpgr, 1, 0, 0)  # fit an AR(1)
```

```
## initial  value -4.589567
## iter   2 value -4.654150
## iter   3 value -4.654150
## iter   4 value -4.654151
## iter   4 value -4.654151
## iter   4 value -4.654151
```

```
## final  value -4.654151
## converged
## initial  value -4.655919
## iter   2 value -4.655921
## iter   3 value -4.655922
## iter   4 value -4.655922
## iter   5 value -4.655922
## iter   5 value -4.655922
## iter   5 value -4.655922
## final  value -4.655922
## converged
```




```
acf2(resid(u$fit)^2, 20)          # get (p)acf of the squared residuals
```




```
##      [,1] [,2] [,3] [,4]  [,5] [,6]  [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  0.12 0.13 0.03 0.13  0.01 0.05 -0.03 0.06 0.08 -0.08  0.09  0.10  0.01
## PACF 0.12 0.12 0.00 0.12 -0.02 0.02 -0.04 0.05 0.08 -0.12  0.11  0.09 -0.05
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20]
## ACF   0.04  0.15 -0.02  0.04 -0.05  0.01  0.05
## PACF  0.05  0.13 -0.09  0.01 -0.05  0.00  0.04
```

ACF and PACF of the squares of the residuals from the AR(1) fit on U.S. GNP. It appears that there may be some dependence, albeit small, left in the residuals.

If the GNP noise term is ARCH, the squares of the residuals from the fit should behave like a non-Gaussian AR(1) process

```
library(fGarch)
```

```
## NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
## attached to the search() path when 'fGarch' is attached.
##
## If needed attach them yourself in your R script by e.g.,
##         require("timeSeries")
```

```
summary(garchFit(~arma(1,0)+garch(1,0), gnpgr))
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(1, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 0)
##  ARMA Order:                1 0
##  Max ARMA Order:            1
##  GARCH Order:               1 0
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          norm
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          222
##  Recursion Init:            mci
##  Series Scale:              0.01015924
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                       U          V     params includes
##     mu      -8.20681904   8.206819 0.8205354     TRUE
##     ar1     -0.99999999   1.000000 0.3466459     TRUE
##     omega    0.00000100 100.000000 0.1000000     TRUE
##     alpha1   0.00000001   1.000000 0.1000000     TRUE
##     gamma1  -0.99999999   1.000000 0.1000000    FALSE
##     delta    0.00000000   2.000000 2.0000000    FALSE
##     skew     0.10000000  10.000000 1.0000000    FALSE
##     shape    1.00000000  10.000000 4.0000000    FALSE
##  Index List of Parameters to be Optimized:
##     mu    ar1  omega alpha1
##      1      2      3      4
##  Persistence:                  0.1
##
##
```

```
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     682.89527: 0.820535 0.346646 0.100000 0.100000
##   1:     308.43148: 0.763492 0.258112  1.06104 0.352453
##   2:     306.07332: 0.681276 0.195897  1.04763 0.304072
##   3:     301.00807: 0.561958 0.448458 0.825277 0.0402737
##   4:     298.88361: 0.383716 0.465477 0.632947 0.385969
##   5:     296.74288: 0.504144 0.389445 0.683634 0.247795
##   6:     296.67703: 0.497724 0.366843 0.688130 0.229496
##   7:     296.60039: 0.500011 0.385702 0.703145 0.211105
##   8:     296.59692: 0.515645 0.374174 0.690079 0.194961
##   9:     296.56381: 0.513570 0.367018 0.702272 0.200013
##  10:     296.55723: 0.523440 0.363126 0.708406 0.194151
##  11:     296.55632: 0.522578 0.364913 0.710104 0.194839
##  12:     296.55598: 0.520871 0.364956 0.710924 0.193212
##  13:     296.55568: 0.519486 0.366571 0.710213 0.194510
##  14:     296.55568: 0.519509 0.366597 0.710266 0.194512
##  15:     296.55568: 0.519511 0.366586 0.710290 0.194452
##  16:     296.55568: 0.519505 0.366562 0.710299 0.194464
##  17:     296.55568: 0.519526 0.366560 0.710295 0.194472
##  18:     296.55568: 0.519522 0.366563 0.710295 0.194471
##
## Final Estimate of the Negative LLH:
##  LLH:  -722.2849    norm LLH:  -3.253536
##           mu          ar1        omega       alpha1
## 0.0052779470 0.3665625602 0.0000733096 0.1944713367
##
## R-optimhess Difference Approximated Hessian Matrix:
##                  mu          ar1         omega        alpha1
## mu     -2749495.405 -24170.124893  4.546826e+06 -1.586692e+03
## ar1      -24170.125   -390.266820  1.253879e+04 -6.733791e+00
## omega   4546825.987  12538.785675 -1.590043e+10 -7.069341e+05
## alpha1    -1586.692     -6.733791 -7.069341e+05 -1.425395e+02
## attr(,"time")
## Time difference of 0.002082109 secs
##
## --- END OF TRACE ---
```

```
##
##
## Time to Estimate Parameters:
##  Time difference of 0.010741 secs
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(1, 0) + garch(1, 0), data = gnpgr)
##
## Mean and Variance Equation:
##  data ~ arma(1, 0) + garch(1, 0)
## <environment: 0x1212f3a18>
##  [data = gnpgr]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu         ar1       omega      alpha1
## 0.00527795  0.36656256  0.00007331  0.19447134
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     5.278e-03   8.996e-04    5.867 4.44e-09 ***
## ar1    3.666e-01   7.514e-02    4.878 1.07e-06 ***
## omega  7.331e-05   9.011e-06    8.135 4.44e-16 ***
## alpha1 1.945e-01   9.554e-02    2.035   0.0418 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  722.2849    normalized:  3.253536
##
## Description:
##  Fri May  5 11:14:26 2023 by user:
```

```
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  9.118036  0.01047234
##  Shapiro-Wilk Test  R    W      0.9842406 0.0143365
##  Ljung-Box Test     R    Q(10)  9.874326  0.4515875
##  Ljung-Box Test     R    Q(15)  17.55855  0.2865844
##  Ljung-Box Test     R    Q(20)  23.41363  0.2689437
##  Ljung-Box Test     R^2  Q(10)  19.2821   0.03682245
##  Ljung-Box Test     R^2  Q(15)  33.23648  0.004352735
##  Ljung-Box Test     R^2  Q(20)  37.74259  0.009518988
##  LM Arch Test       R    TR^2   25.41625  0.01296901
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -6.471035 -6.409726 -6.471669 -6.446282
```

We obtain $\varphi_0 = .005$ (called μ in the output) and $\varphi_1 = .367$ (called $ar1$) for the AR(1) parameter estimates; in Example 3.38 the values were .005 and .347, respectively.

The ARCH(1) parameter estimates are $\hat{\alpha}_0 = 0$ (called `omega`) for the constant and $\hat{\alpha}_1 = .195$, which is significant with a p-value of about .04.

There are a number of tests that are performed on the residuals [`R`] or the squared residuals [`R^2`].

For example, the Jarque–Bera statistic tests the residuals of the fit for normality based on the observed skewness and kurtosis, and it appears that the residuals have some non-normal skewness and kurtosis.

The Shapiro–Wilk statistic tests the residuals of the fit for normality based on the empirical order statistics.

3.2 Example 5.5

The daily returns of the NYSE exhibit classic GARCH features.

The GARCH(1,1) model admits a non-Gaussian ARMA(1, 1) model for the squared process.

```r
library(fGarch)
summary(nyse.g <- garchFit(~garch(1,1), nyse))
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(0, 0)
```

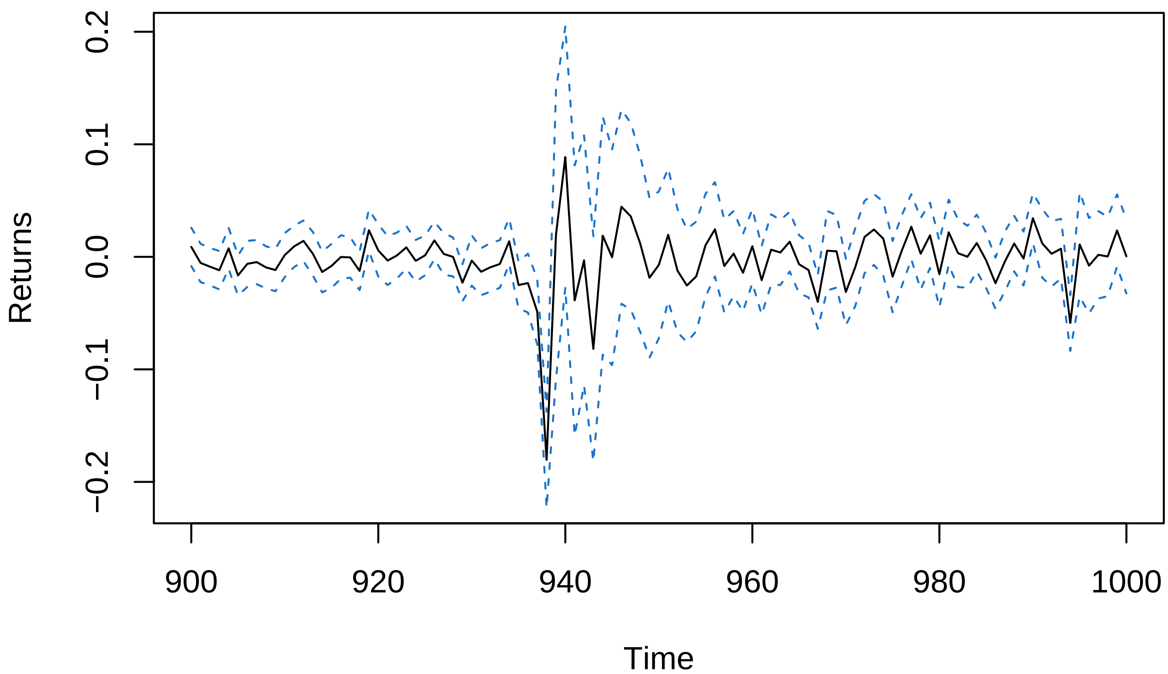
```
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 1)
##  ARMA Order:                0 0
##  Max ARMA Order:            0
##  GARCH Order:               1 1
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          norm
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          2000
##  Recursion Init:            mci
##  Series Scale:              0.009862128
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                       U           V     params includes
##     mu     -0.47751157   0.4775116 0.04775116     TRUE
##     omega   0.00000100 100.0000000 0.10000000     TRUE
##     alpha1  0.00000001   1.0000000 0.10000000     TRUE
##     gamma1 -0.99999999   1.0000000 0.10000000    FALSE
##     beta1   0.00000001   1.0000000 0.80000000     TRUE
##     delta   0.00000000   2.0000000 2.00000000    FALSE
##     skew    0.10000000  10.0000000 1.00000000    FALSE
##     shape   1.00000000  10.0000000 4.00000000    FALSE
##  Index List of Parameters to be Optimized:
##     mu  omega alpha1  beta1
##      1      2      3      5
##  Persistence:                  0.9
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     2530.4637: 0.0477512 0.100000 0.100000 0.800000
```

```
##   1:     2519.8035: 0.0477528 0.0890814 0.0971176 0.792299
##   2:     2518.2039: 0.0477610 0.0778289 0.104356 0.789508
##   3:     2517.1996: 0.0477746 0.0798531 0.116900 0.793679
##   4:     2516.7331: 0.0478516 0.0691264 0.124616 0.792366
##   5:     2516.2421: 0.0481458 0.0683125 0.118305 0.802390
##   6:     2516.2194: 0.0482907 0.0677374 0.114458 0.803568
##   7:     2516.1678: 0.0485045 0.0683114 0.114028 0.805840
##   8:     2516.1316: 0.0487284 0.0669777 0.113452 0.807117
##   9:     2515.9510: 0.0525311 0.0734963 0.118200 0.793747
##  10:     2515.4092: 0.0634778 0.0698334 0.119845 0.800191
##  11:     2515.3030: 0.0664530 0.0680584 0.112250 0.803959
##  12:     2515.2868: 0.0674938 0.0694502 0.111239 0.806769
##  13:     2515.2002: 0.0670153 0.0686849 0.114810 0.803988
##  14:     2515.1933: 0.0675394 0.0680343 0.114069 0.804194
##  15:     2515.1731: 0.0680649 0.0682215 0.113932 0.804775
##  16:     2515.1629: 0.0685867 0.0674527 0.114833 0.805594
##  17:     2515.1136: 0.0724613 0.0662774 0.112972 0.808527
##  18:     2515.1033: 0.0750936 0.0676551 0.114639 0.804962
##  19:     2515.1021: 0.0747415 0.0672663 0.114077 0.806071
##  20:     2515.1021: 0.0747241 0.0672569 0.114080 0.806079
##  21:     2515.1021: 0.0747252 0.0672580 0.114080 0.806077
##
## Final Estimate of the Negative LLH:
##  LLH:  -6723.005    norm LLH:  -3.361502
##           mu        omega       alpha1        beta1
## 7.369498e-04 6.541615e-06 1.140803e-01 8.060773e-01
##
## R-optimhess Difference Approximated Hessian Matrix:
##                   mu         omega        alpha1         beta1
## mu     -3.143305e+07 -3.035042e+08  7.493639e+03     -17309.08
## omega  -3.035042e+08 -7.277616e+12 -2.769132e+08 -450073513.64
## alpha1  7.493639e+03 -2.769132e+08 -2.089060e+04     -21533.02
## beta1  -1.730908e+04 -4.500735e+08 -2.153302e+04     -30843.19
## attr(,"time")
## Time difference of 0.006901979 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
```

```
##  Time difference of 0.09819484 secs
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 1), data = nyse)
##
## Mean and Variance Equation:
##  data ~ garch(1, 1)
## <environment: 0x125e38000>
##  [data = nyse]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1       beta1
## 7.3695e-04  6.5416e-06  1.1408e-01  8.0608e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     7.369e-04   1.786e-04    4.126 3.69e-05 ***
## omega  6.542e-06   1.455e-06    4.495 6.94e-06 ***
## alpha1 1.141e-01   1.604e-02    7.114 1.13e-12 ***
## beta1  8.061e-01   2.973e-02   27.112  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  6723.005    normalized:  3.361502
##
## Description:
##  Fri May  5 11:14:26 2023 by user:
##
##
## Standardised Residuals Tests:
```

```
##                                    Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  3628.415  0
##  Shapiro-Wilk Test  R    W      0.9515562 0
##  Ljung-Box Test     R    Q(10)  29.69242  0.0009616813
##  Ljung-Box Test     R    Q(15)  30.50938  0.01021164
##  Ljung-Box Test     R    Q(20)  32.81143  0.03538324
##  Ljung-Box Test     R^2  Q(10)  3.510505  0.9667405
##  Ljung-Box Test     R^2  Q(15)  4.408852  0.9960585
##  Ljung-Box Test     R^2  Q(20)  6.68935   0.9975864
##  LM Arch Test       R    TR^2   3.967784  0.9840107
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -6.719005 -6.707803 -6.719013 -6.714891
```

```
u = nyse.g@sigma.t
plot(window(nyse, start=900, end=1000), ylim=c(-.22,.2), ylab="NYSE
Returns")
lines(window(nyse-2*u, start=900, end=1000), lty=2, col=4)
lines(window(nyse+2*u, start=900, end=1000), lty=2, col=4)
```



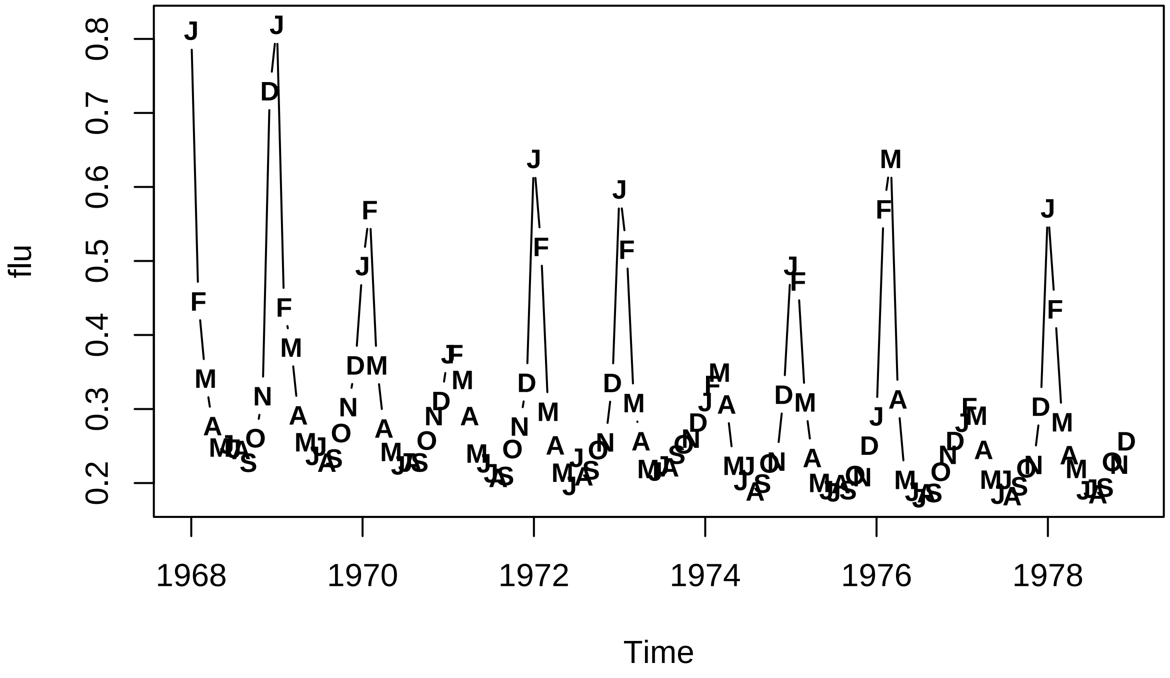
We calculated and plotted the 100 observations from the middle of the data (which includes the October 19, 1987 crash) along with the one-step-ahead predictions of the corresponding volatility, σ_t^2. The results are displayed as the data $\pm 2\hat{\sigma}_t$ as a dashed line surrounding the data

4 Section 5.5 Threshold Models

Many approaches to modeling nonlinear series exist. Here, we focus on the class of threshold autoregressive models. The basic idea of these models is that of fitting local linear AR(p) models.

4.1 Example 5.6

```
# Plot data with month initials as points
plot(flu, type="c")
Months = c("J","F","M","A","M","J","J","A","S","O","N","D")
points(flu, pch=Months, cex=.8, font=2)
```




```
# clearly, the data is nonlinear
```

The process has a possibility of jumping to a large number once $x_t = flu_t - flu_{t-1}$ exceeds about .05.

We fit the following threshold models.

$$x_t = \alpha^{(1)} + \sum_{j=1}^{p} \phi_j^{(1)} x_{t-j} + w_t^{(1)}, x_{t-1} < .05 \tag{2}$$

$$x_t = \alpha^{(2)} + \sum_{j=1}^{p} \phi_j^{(2)} x_{t-j} + w_t^{(2)}, x_{t-1} \geq .05 \tag{3}$$

with p=6, assuming this would be larger than necessary.

```
# Start analysis
dflu = diff(flu)
thrsh = .05 # threshold
Z = ts.intersect(dflu, lag(dflu,-1), lag(dflu,-2), lag(dflu,-3),
lag(dflu,-4))
ind1 = ifelse(Z[,2] < thrsh, 1, NA) # indicator < thrsh
ind2 = ifelse(Z[,2] < thrsh, NA, 1) # indicator >= thrsh

# expect jump in next period if previous period exceeded thrsh

X1 = Z[,1]*ind1
X2 = Z[,1]*ind2
summary(fit1<-lm(X1~Z[,2:5])) # case 1
```

```
##
## Call:
## lm(formula = X1 ~ Z[, 2:5])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13312 -0.02049  0.00218  0.01667  0.26666
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.004471   0.004894   0.914 0.363032
## Z[, 2:5]lag(dflu, -1)  0.506650   0.078319   6.469  3.2e-09 ***
## Z[, 2:5]lag(dflu, -2) -0.200086   0.056573  -3.537 0.000604 ***
## Z[, 2:5]lag(dflu, -3)  0.121047   0.054463   2.223 0.028389 *
## Z[, 2:5]lag(dflu, -4) -0.110938   0.045979  -2.413 0.017564 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.04578 on 105 degrees of freedom
##   (17 observations deleted due to missingness)
## Multiple R-squared:  0.3763, Adjusted R-squared:  0.3526
## F-statistic: 15.84 on 4 and 105 DF,  p-value: 3.568e-10
```

```
summary(fit2<-lm(X2~Z[,2:5])) # case 2
```

```
##
## Call:
## lm(formula = X2 ~ Z[, 2:5])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.089975 -0.036825 -0.006328  0.040765  0.129509
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.40794    0.04675   8.726 1.53e-06 ***
## Z[, 2:5]lag(dflu, -1) -0.74833    0.16644  -4.496 0.000732 ***
## Z[, 2:5]lag(dflu, -2) -1.03231    0.21137  -4.884 0.000376 ***
## Z[, 2:5]lag(dflu, -3) -2.04504    1.05000  -1.948 0.075235 .
## Z[, 2:5]lag(dflu, -4) -6.71178    1.24538  -5.389 0.000163 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0721 on 12 degrees of freedom
##   (110 observations deleted due to missingness)
## Multiple R-squared:  0.9207, Adjusted R-squared:  0.8943
## F-statistic: 34.85 on 4 and 12 DF,  p-value: 1.618e-06
```

An order p = 4 was finally selected for each part of the model. The final model was

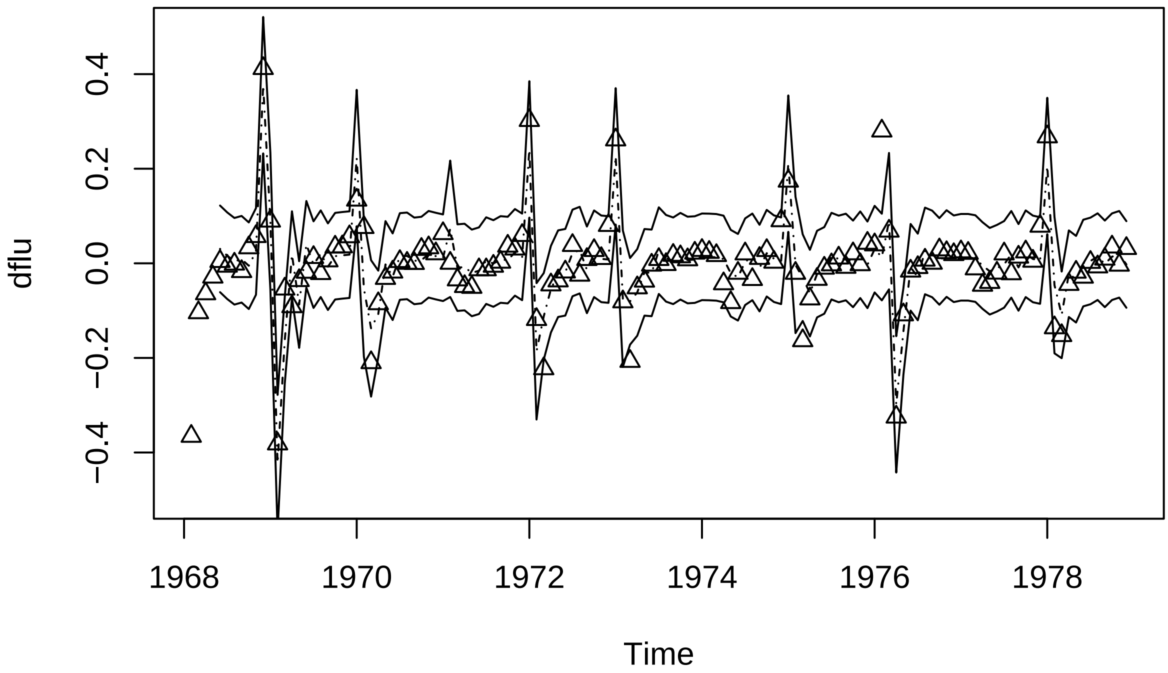
$$\hat{x}_t = .51_{(.08)}x_{t-1} - .20_{(.06)}x_{t-2} + .12_{(.05)}x_{t-3} - .11_{(.05)}x_{t-4} + \hat{w}_t^{(1)}, x_{t-1} < .05 \quad (4)$$

$$\hat{x}_t = .40 - .75_{(.17)}x_{t-1} - 1.03_{(.21)}x_{t-2} - 2.05_{(1.05)}x_{t-3} - 6.71_{(1.25)}x_{t-4} + \hat{w}_t^{(2)}, x_{t-1} \geq .05 \quad (5)$$

with $\hat{\sigma}_1 = .05, \hat{\sigma}_2 = .07$

```
D = cbind(rep(1, nrow(Z)), Z[,2:5]) # get predictions
b1 = fit1$coef
```

```
b2 = fit2$coef
p1 = D%*%b1
p2 = D%*%b2
prd = ifelse(Z[,2] < thrsh, p1, p2)
plot(dflu, type="p", pch=2, ylim=c(-.5,.5))
lines(prd, lty=4)

# prediction error
prde1 = sqrt(sum(resid(fit1)^2)/df.residual(fit1))
prde2 = sqrt(sum(resid(fit2)^2)/df.residual(fit2))
prde = ifelse(Z[,2] < thrsh, prde1, prde2)
lines(prd + 2*prde)
lines(prd - 2*prde)
```



First differenced U.S. monthly pneumonia and influenza deaths (triangles); one-month-ahead predictions (dashed line) with ± 2 prediction error bounds (solid line).

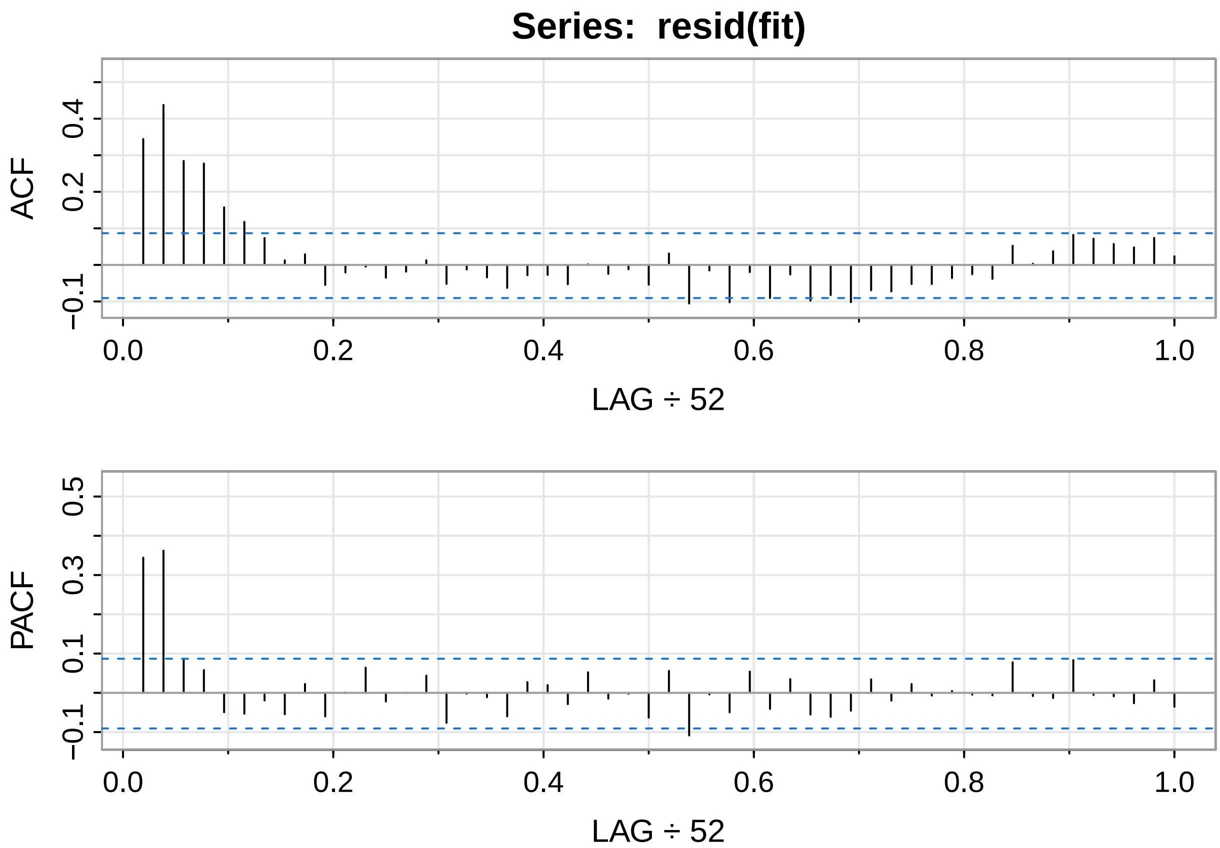
5 Section 5.6 Regression with Autocorrelated Errors

5.1 Example 5.7

consider the regression model

$$M_t = \beta_1 + \beta_2 t + \beta_3 T_t + \beta_4 T_t^2 + \beta_5 P_t + x_t \tag{6}$$

```
trend = time(cmort)
temp = tempr - mean(tempr)
temp2 = temp^2
fit = lm(cmort~trend + temp + temp2 + part, na.action=NULL)
acf2(resid(fit), 52) # implies AR2
```




```
##       [,1] [,2] [,3] [,4]  [,5]  [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  0.34 0.44 0.28 0.28  0.16  0.12  0.07  0.01 0.03 -0.05 -0.02  0.00 -0.04
## PACF 0.34 0.36 0.08 0.06 -0.05 -0.05 -0.02 -0.05 0.02 -0.06  0.00  0.06 -0.02
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF  -0.02  0.01 -0.05 -0.01 -0.03 -0.06 -0.03 -0.03 -0.05  0.00 -0.02 -0.01
## PACF  0.00  0.04 -0.08  0.00 -0.01 -0.06  0.03  0.02 -0.03  0.05 -0.01  0.00
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
## ACF  -0.05  0.03 -0.11 -0.02 -0.10 -0.02 -0.09 -0.03 -0.10 -0.08 -0.10 -0.07
## PACF -0.06  0.06 -0.11  0.00 -0.05  0.05 -0.04  0.04 -0.06 -0.06 -0.05  0.03
##      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49]
## ACF  -0.07 -0.05 -0.05 -0.04 -0.03 -0.04  0.05  0.00  0.04  0.08  0.07  0.06
## PACF -0.02  0.02 -0.01  0.00  0.00 -0.01  0.08 -0.01 -0.01  0.08 -0.01 -0.01
##      [,50] [,51] [,52]
```

```
## ACF   0.05  0.07  0.02
## PACF -0.03  0.03 -0.04
```

The graph suggest an AR(2) model for the residuals. Thus, x_t is AR(2),

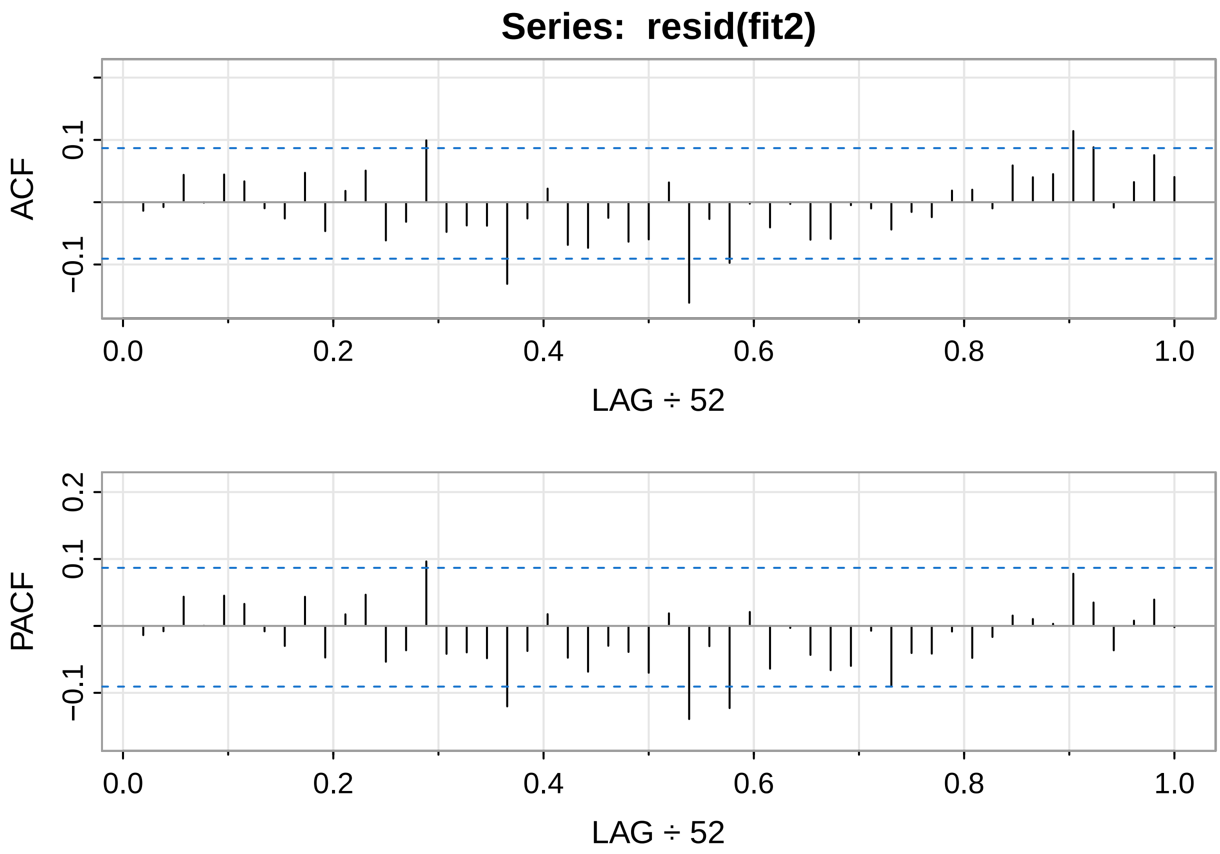
$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + w_t \tag{7}$$

and w_t is white noise

```
fit2 = arima(cmort, order=c(2,0,0),
             xreg=cbind(trend,temp,temp2,part))
fit2
```

```
##
## Call:
## arima(x = cmort, order = c(2, 0, 0), xreg = cbind(trend, temp, temp2, part))
##
## Coefficients:
##          ar1     ar2  intercept    trend     temp   temp2    part
##       0.3848  0.4326  3075.1482  -1.5165  -0.0190  0.0154  0.1545
## s.e.  0.0436  0.0400   834.7233   0.4226   0.0495  0.0020  0.0272
##
## sigma^2 estimated as 26.01:  log likelihood = -1549.04,  aic = 3114.07
```

```
acf2(resid(fit2), 52) # no obvious departures from whiteness
```




```
##       [,1]  [,2] [,3] [,4] [,5] [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  -0.01 -0.01 0.04    0 0.04 0.03 -0.01 -0.03 0.05 -0.05  0.02  0.05 -0.06
## PACF -0.01 -0.01 0.04    0 0.05 0.03 -0.01 -0.03 0.04 -0.05  0.02  0.05 -0.05
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF  -0.03   0.1 -0.05 -0.04 -0.04 -0.13 -0.03  0.02 -0.07 -0.07 -0.03 -0.06
## PACF -0.04   0.1 -0.04 -0.04 -0.05 -0.12 -0.04  0.02 -0.05 -0.07 -0.03 -0.04
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
## ACF  -0.06  0.03 -0.16 -0.03 -0.10  0.00 -0.04     0 -0.06 -0.06  0.00 -0.01
## PACF -0.07  0.02 -0.14 -0.03 -0.12  0.02 -0.06     0 -0.04 -0.07 -0.06 -0.01
##      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49]
## ACF  -0.04 -0.02 -0.02  0.02  0.02 -0.01  0.06  0.04  0.05  0.11  0.09 -0.01
## PACF -0.09 -0.04 -0.04 -0.01 -0.05 -0.02  0.02  0.01  0.00  0.08  0.03 -0.04
##      [,50] [,51] [,52]
## ACF   0.03  0.08  0.04
## PACF  0.01  0.04  0.00
```

The sample ACF and PACF of the residuals show no obvious departure from whiteness.

```
Q = Box.test(resid(fit2), 12, type="Ljung")$statistic # = 6.91
Q
```

```
## X-squared
##  6.913324
```

```
pchisq(as.numeric(Q), df=(12-2), lower=FALSE) # p-value = .73
```

```
## [1] 0.7336016
```

The Q-statistic supports the whiteness of the residuals (note the residuals are from an AR(2) fit, and this is used in the calculation of the p-value). We also note that the trend is no longer significant, and we may wish to rerun the analysis with the trend removed.

6 Section 5.7 Lagged Regression: Transfer Function Modeling

Consider the lagged regression model

$$y_t = \sum_{j=0}^{\infty} \alpha_j x_{t-j} + \eta_t = \alpha(B)x_t + \eta_t \tag{8}$$

We assign the ARMA(p,q) modeling on the input and noise seris, i.e.,

$$\phi(B)x_t = \theta(B)w_t \tag{9}$$
$$\phi_\eta(B)\eta_t = \theta_\eta(B)z_t \tag{10}$$

where w_t and z_t are independent white noise with variances σ_w^2, σ_z^2.

Hence, $w_t = \frac{\phi(B)}{\theta(B)}x_t$ and $z_t = \frac{\phi_\eta(B)}{\theta_\eta(B)}\eta_t$

So, we have $\tilde{y}_t = \frac{\phi(B)}{\theta(B)}y_t = \alpha(B)w_t + \frac{\phi(B)}{\theta(B)}\eta_t = \alpha(B)w_t + \tilde{\eta}_t$

The series w_t is a prewhitened version of the input series, and its cross-correlation with the transformed output series $\tilde{y}_t$ will be just $\gamma_{\tilde{y}w}(h) = \sigma_w^2\alpha_h$

By computing the cross-correlation between the prewhitened input series and the transformed output series should yield a rough estimate of the behavior of $\alpha(B)$

It turns out, $\alpha(B) = \frac{\delta(B)B^d}{\omega(B)}$

Then $y_t = \frac{\delta(B)B^d}{\omega(B)}x_t + \eta_t$

or $\omega(B)y_t = \delta(B)B^d x_t + \omega(B)\eta_t$

Let $u_t = \omega(B)\eta_t$

General steps:

(i) Fit an ARMA model to the input series x_t to estimate the parameters in $\phi(B)x_t = \theta(B)w_t$. Retain ARMA coefficients for use in step (ii) and the fitted residuals $\hat{w}_t$ for use in Step (iii).

(ii) Apply the operator determined in step (i) to the output series y_t, that is, $\tilde{y}_t = \frac{\hat{\phi}(B)}{\hat{\theta}(B)} y_t$, to determine the transformed output series $\tilde{y}_t$.

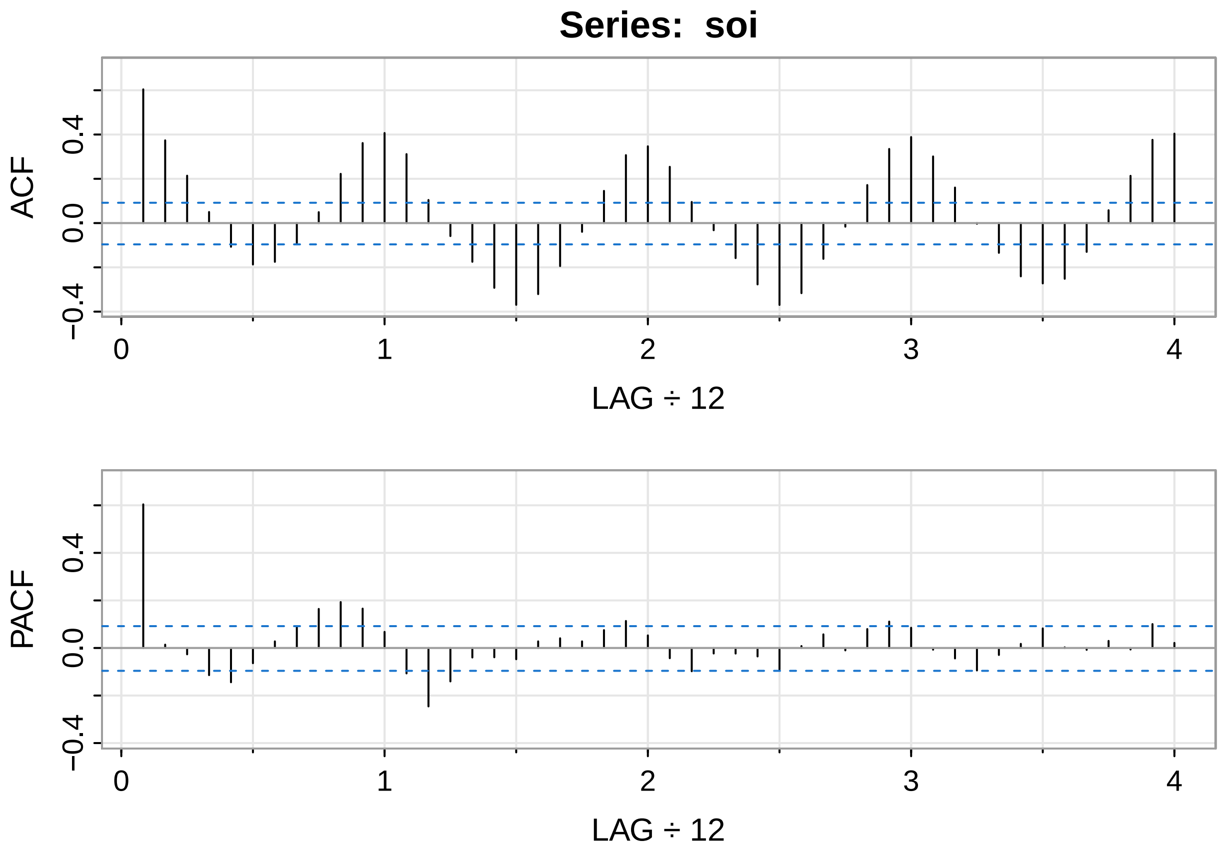
(iii) Use the cross-correlation function between $\tilde{y}_t$ and $\hat{w}_t$ in steps (i) and (ii) to suggest a form for the components of the polynomial $\alpha(B) = \frac{\delta(B)B^d}{\omega(B)}$ and the estimated time delay d.

(iv) Obtain $\hat{\beta} = (\hat{\omega}_1, ..., \hat{\omega}_r, \hat{\delta}_0, \hat{\delta}_1, .., \hat{\delta}_s)$ by fitting a linear regression of the form $\omega(B)y_t = \delta(B)B^d x_t + \omega(B)\eta_t$. Retain the residuals $\hat{u}_t$ for use in step (v)

(v) Apply the moving average transformation $u_t = \omega(B)\eta_t$ to the residuals $\hat{u}_t$ to find the noise series $\hat{\eta}_t$ and fit an ARMA model to the noise, obtaining the estimated coefficients in $\hat{\phi}_\eta(B)$ and $\hat{\theta}_\eta(B)$

6.1 Recruitment Example

```
# Sample ACF and PACF of SOI.
acf2(soi)
```




```
##       [,1] [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF    0.6 0.37  0.21  0.05 -0.11 -0.19 -0.18 -0.10 0.05  0.22  0.36  0.41  0.31
## PACF   0.6 0.01 -0.03 -0.11 -0.14 -0.06  0.03  0.09 0.16  0.19  0.17  0.07 -0.11
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF    0.10 -0.06 -0.17 -0.29 -0.37 -0.32 -0.19 -0.04  0.15  0.31  0.35  0.25
```

```
## PACF -0.25 -0.14 -0.04 -0.04 -0.05  0.03  0.04  0.03  0.08  0.11  0.05 -0.04
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
## ACF    0.1 -0.03 -0.16 -0.28 -0.37 -0.32 -0.16 -0.02  0.17  0.33  0.39  0.30
## PACF  -0.1 -0.02 -0.02 -0.04 -0.09  0.01  0.06 -0.01  0.08  0.11  0.09 -0.01
##      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
## ACF   0.16  0.00 -0.13 -0.24 -0.27 -0.25 -0.13  0.06  0.21  0.38  0.40
## PACF -0.04 -0.09 -0.03  0.02  0.08  0.00 -0.01  0.03 -0.01  0.10  0.02
```

It is clear, from the PACF, that an autoregressive series with p = 1 will do a reasonable job.

```
# step (i)
# Fit an ARMA to the input series
fit = arima(soi, xreg=time(soi), order=c(1, 0, 0))
fit
```

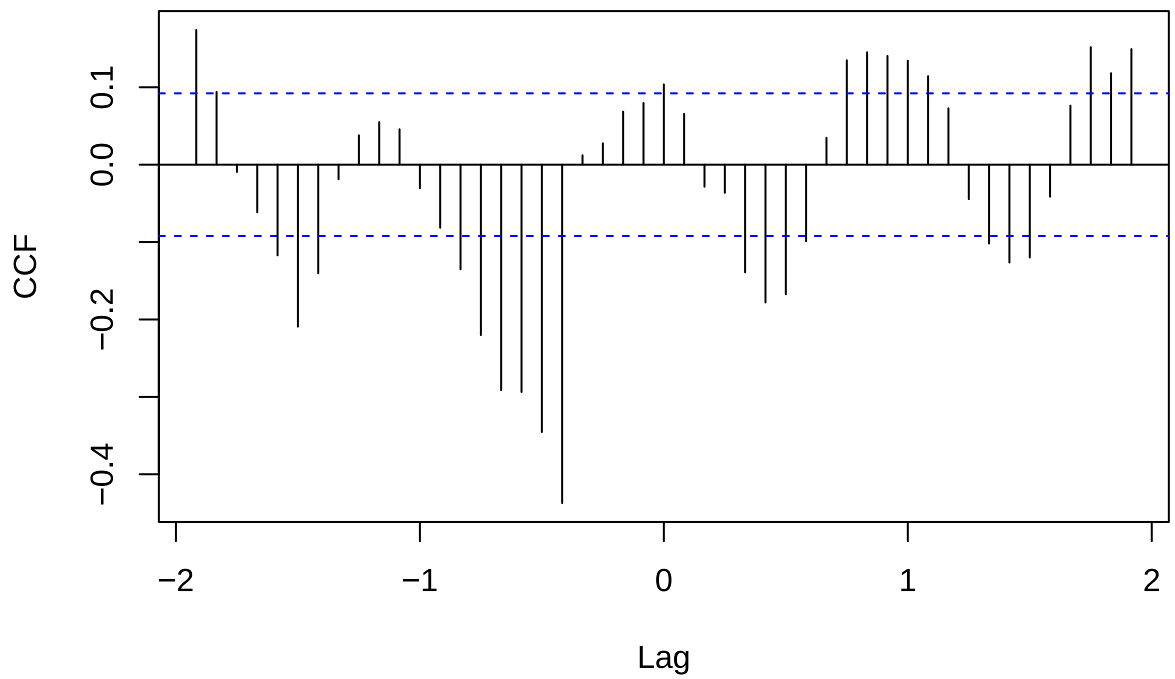
```
##
## Call:
## arima(x = soi, order = c(1, 0, 0), xreg = time(soi))
##
## Coefficients:
##          ar1  intercept  time(soi)
##       0.5875    13.7507    -0.0069
## s.e.  0.0379     6.1775     0.0031
##
## sigma^2 estimated as 0.09181:  log likelihood = -102.1,  aic = 212.19
```

```
# save the ARMA coef
ar1 <- as.numeric(fit$coef[1]) # = 0.5875387

# save the fitted residuals
# the w hat
soi.pw = resid(fit) # prewhitened detrended SOI series
```

Fitting the series gave $\hat{\phi} = .588$ with $\hat{\sigma}_w^2 = .092$, i.e., $(1 - .588B)x_t = w_t$ and we applied the operator $(1 - .588B)$ to both x_t and y_t and computed the cross-correlation function.

```
# step (ii)

# the detrended Recruitment series
rec.d = resid(lm(rec~time(rec), na.action=NULL))
```

```
# the filtered, detrended Recruitment series
# the y tilde, transformed output series
rec.fil = filter(rec.d, filter=c(1, -ar1), method="conv", sides=1)

# step (iii)

# na.action=na.omit is used because rec.fil[1] is NA.
# Noting the apparent shift of d = 5 months and the decrease thereafter
ccf(soi.pw, rec.fil, main="", ylab="CCF", na.action=na.omit)
```



Sample CCF of the prewhitened, detrended SOI and the similarly transformed Recruitment series; negative lags indicate that SOI leads Recruitment.

It seems plausible to hypothesize a model of the form

$$\alpha(B) = \delta_0 B^5(1 + \omega_1 B + \omega_1^2 B^2 + ...) = \frac{\delta_0 B^5}{1 - \omega_1 B} \tag{11}$$

Now we go to step (iv).

```
# step (iv)
rec.d = resid(lm(rec~time(rec), na.action=NULL))
soi.d = resid(lm(soi~time(soi)))
```

```
fish = ts.intersect(rec.d, rec.d1=lag(rec.d,-1), soi.d5=lag(soi,-5),
      dframe=TRUE)

summary(fish.fit <- lm(rec.d~0+rec.d1+soi.d5, data=fish))
```

```
##
## Call:
## lm(formula = rec.d ~ 0 + rec.d1 + soi.d5, data = fish)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.441  -2.492   1.806   6.052  29.932
##
## Coefficients:
##        Estimate Std. Error t value Pr(>|t|)
## rec.d1   0.8531     0.0144   59.23   <2e-16 ***
## soi.d5 -18.8627     1.0038  -18.79   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.02 on 446 degrees of freedom
## Multiple R-squared:  0.9145, Adjusted R-squared:  0.9141
## F-statistic:  2386 on 2 and 446 DF,  p-value: < 2.2e-16
```

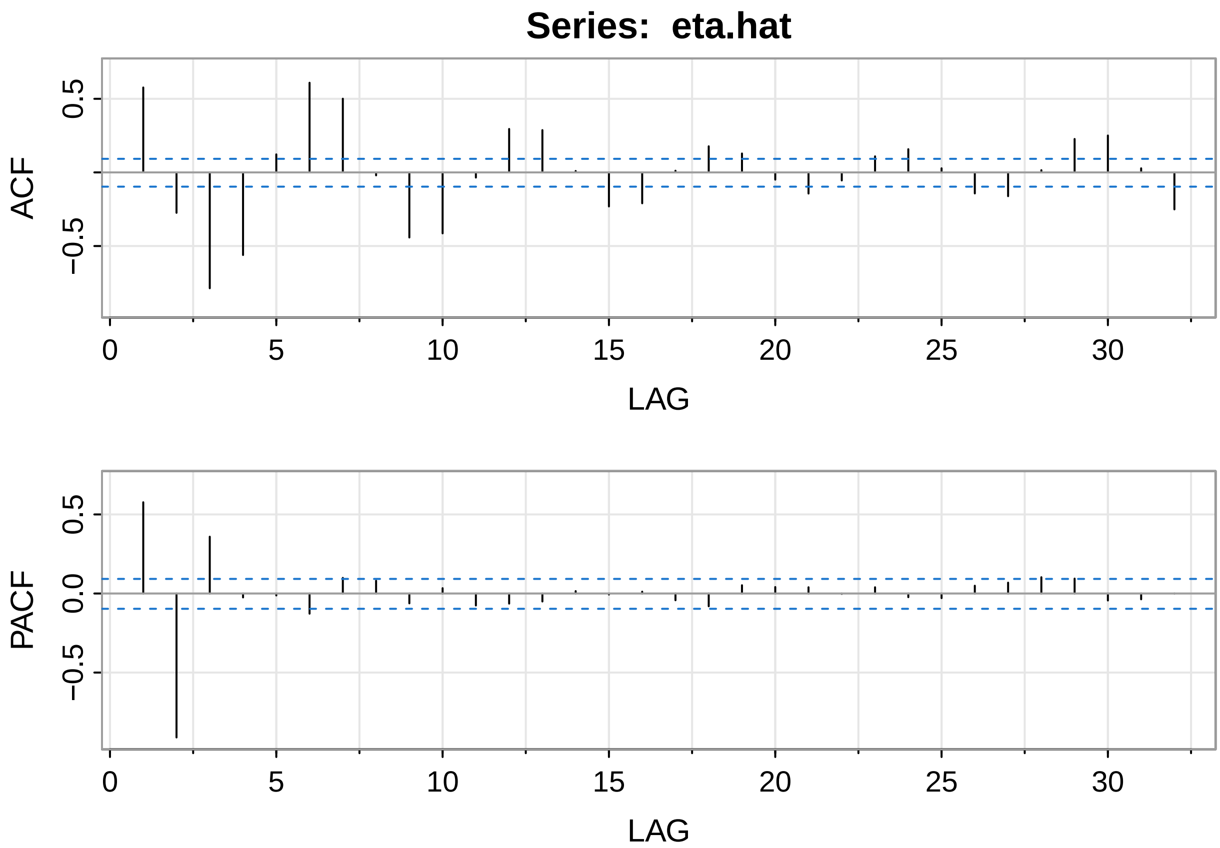
Run the regression $y_t = \omega_1 y_{t-1} + \delta_0 x_{t-5} + u_t$ yielding $\hat{\omega}_1 - .853, \hat{\delta}_0 = -18.86$ where the residuals satisfy $\hat{u}_t = (1 - .853B)\eta_t$

This completes step (iv).

```
# step(v)

om1 = as.numeric(fish.fit$coef[1])

# eta hat
eta.hat = filter(resid(fish.fit), filter=c(1,-om1), method="recur",
sides=1)
```

To complete the specification, we apply the moving average operator above to the residuals $\hat{u}_t$, this help to estimate the original noise series η_t

```
acf2(eta.hat)
```




```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6] [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  0.58 -0.27 -0.79 -0.56  0.12  0.61  0.5 -0.02 -0.44 -0.41 -0.04  0.30
## PACF 0.58 -0.91  0.36 -0.02 -0.01 -0.13  0.1  0.08 -0.06  0.03 -0.08 -0.06
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF   0.29  0.01 -0.23 -0.21  0.01  0.18  0.13 -0.05 -0.14 -0.05  0.11  0.16
## PACF -0.05  0.02 -0.01  0.01 -0.04 -0.08  0.05  0.04  0.04  0.00  0.04 -0.02
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
## ACF   0.03 -0.14 -0.16  0.02  0.23  0.25  0.03 -0.25
## PACF -0.03  0.05  0.07  0.10  0.09 -0.04 -0.04  0.00
```

Lastly, fit a second-order autoregressive model, based on the ACF and PACF.

```
eta.fit <- arima(eta.hat, order=c(3,0,0))
eta.fit
```

```
##
## Call:
## arima(x = eta.hat, order = c(3, 0, 0))
##
## Coefficients:
```

```
##          ar1      ar2     ar3  intercept
##       1.4261  -1.3017  0.3557     1.6015
## s.e.  0.0441   0.0517  0.0441     0.6483
##
## sigma^2 estimated as 50.85:  log likelihood = -1517.91,  aic = 3045.82
```

We obtain $(1 - 1.426B + 1.30B^2 - .356B^3)\eta_t = z_t$, with $\hat{\sigma}_z^2 = 50.85$ as the estimated error variance

7 Section 5.8 Multivariate ARMAX Models

$\mathbf{x}_t = \Gamma \mathbf{u}_t + \sum_{j=1}^{p} \Phi_j \mathbf{x}_{t-j} + \mathbf{w}_j$

where Γ is a p x r parameter matrix. The X in ARX refers to the exogenous vector process we have denoted here by $\mathbf{u}_t$. The introduction of exogenous variables through replacing □ by $\Gamma \mathbf{u}_t$ does not present any special problems in making inferences and we will often drop the X for being superfluous.

7.1 Example 5.10

VAR(1)

Consider cardiovascular mortality x_{t1}, temperature x_{t2}, and particulate levels x_{t3}

The VAR(1) model is

$$
\begin{align}
x_{t1} &= \alpha_1 + \beta_1 t + \phi_{11} x_{t-1,1} + \phi_{12} x_{t-1,2} + \phi_{13} x_{t-1,3} + w_{t1} \tag{12} \\
x_{t2} &= \alpha_2 + \beta_2 t + \phi_{21} x_{t-1,1} + \phi_{22} x_{t-1,2} + \phi_{23} x_{t-1,3} + w_{t2} \tag{13} \\
x_{t3} &= \alpha_3 + \beta_3 t + \phi_{31} x_{t-1,1} + \phi_{32} x_{t-1,2} + \phi_{33} x_{t-1,3} + w_{t3} \tag{14}
\end{align}
$$

```
library(astsa)
library(vars)
```

```
## Loading required package: MASS
```

```
## Loading required package: strucchange
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich

## Loading required package: urca

## Loading required package: lmtest
```

```
x=cbind(cmort, tempr, part)
summary(VAR(x, p=1, type="both")) # "both" fits constant + trend
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: cmort, tempr, part
## Deterministic variables: both
## Sample size: 507
## Log Likelihood: -5116.02
## Roots of the characteristic polynomial:
## 0.8931 0.4953 0.1444
## Call:
## VAR(y = x, p = 1, type = "both")
##
##
## Estimation results for equation cmort:
## ======================================
## cmort = cmort.l1 + tempr.l1 + part.l1 + const + trend
##
##            Estimate Std. Error t value Pr(>|t|)
## cmort.l1  0.464824   0.036729  12.656  < 2e-16 ***
## tempr.l1 -0.360888   0.032188 -11.212  < 2e-16 ***
## part.l1   0.099415   0.019178   5.184 3.16e-07 ***
## const    73.227292   4.834004  15.148  < 2e-16 ***
## trend    -0.014459   0.001978  -7.308 1.07e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
```

```
## Residual standard error: 5.583 on 502 degrees of freedom
## Multiple R-Squared: 0.6908,  Adjusted R-squared: 0.6883
## F-statistic: 280.3 on 4 and 502 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation tempr:
## ======================================
## tempr = cmort.l1 + tempr.l1 + part.l1 + const + trend
##
##           Estimate Std. Error t value Pr(>|t|)
## cmort.l1 -0.244046   0.042105  -5.796 1.20e-08 ***
## tempr.l1  0.486596   0.036899  13.187  < 2e-16 ***
## part.l1  -0.127661   0.021985  -5.807 1.13e-08 ***
## const    67.585598   5.541550  12.196  < 2e-16 ***
## trend    -0.006912   0.002268  -3.048  0.00243 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 6.4 on 502 degrees of freedom
## Multiple R-Squared: 0.5007,  Adjusted R-squared: 0.4967
## F-statistic: 125.9 on 4 and 502 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation part:
## =====================================
## part = cmort.l1 + tempr.l1 + part.l1 + const + trend
##
##           Estimate Std. Error t value Pr(>|t|)
## cmort.l1 -0.124775   0.079013  -1.579    0.115
## tempr.l1 -0.476526   0.069245  -6.882 1.77e-11 ***
## part.l1   0.581308   0.041257  14.090  < 2e-16 ***
## const    67.463501  10.399163   6.487 2.10e-10 ***
## trend    -0.004650   0.004256  -1.093    0.275
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 12.01 on 502 degrees of freedom
## Multiple R-Squared: 0.3732,  Adjusted R-squared: 0.3683
```

```
## F-statistic: 74.74 on 4 and 502 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##        cmort  tempr   part
## cmort 31.172  5.975  16.65
## tempr  5.975 40.965  42.32
## part  16.654 42.323 144.26
##
## Correlation matrix of residuals:
##        cmort  tempr   part
## cmort 1.0000 0.1672 0.2484
## tempr 0.1672 1.0000 0.5506
## part  0.2484 0.5506 1.0000
```

Note that t here is `time(cmort)`, which is $1970 + \Delta(t-1)$, where $\Delta = 1/52$ for t=1,…,508, ending at 1979.75

prediction equation for mortality:

$\hat{M}_t = 73.23 - 0.014t + 0.46M_{t-1} - 0.36T_{t-1} + 0.10P_{t-1}$

7.2 Example 5.11

Select a VAR(p) model and then fit the model. The selection criteria used in the package are AIC, HQ, BIC (SC), and Final Prediction Error (FPE).

```
VARselect(x, lag.max = 10, type="both")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      9      5      2      9
##
## $criteria
##                   1           2           3           4           5           6
## AIC(n)     11.73780    11.30185    11.26788    11.23030    11.17634    11.15266
## HQ(n)      11.78758    11.38149    11.37738    11.36967    11.34557    11.35176
## SC(n)      11.86463    11.50477    11.54689    11.58541    11.60755    11.65996
## FPE(n) 125216.91717 80972.28678 78268.19568 75383.73647 71426.10041 69758.25113
##                  7           8           9          10
## AIC(n)    11.15247    11.12878    11.11915    11.12019
```

```
## HQ(n)      11.38144    11.38760    11.40784    11.43874
## SC(n)      11.73587    11.78827    11.85473    11.93187
## FPE(n) 69749.89175 68122.40518 67476.96374 67556.45243
```

Note that BIC picks the order p=2 model. Fitting the model selected by BIC we obtain:

```
summary(fit <- VAR(x, p=2, type="both"))
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: cmort, tempr, part
## Deterministic variables: both
## Sample size: 506
## Log Likelihood: -4987.186
## Roots of the characteristic polynomial:
## 0.8807 0.8807 0.5466 0.4746 0.4746 0.4498
## Call:
## VAR(y = x, p = 2, type = "both")
##
##
## Estimation results for equation cmort:
## ======================================
## cmort = cmort.l1 + tempr.l1 + part.l1 + cmort.l2 + tempr.l2 + part.l2 + const + trend
##
##           Estimate Std. Error t value Pr(>|t|)
## cmort.l1  0.297059   0.043734   6.792 3.15e-11 ***
## tempr.l1 -0.199510   0.044274  -4.506 8.23e-06 ***
## part.l1   0.042523   0.024034   1.769  0.07745 .
## cmort.l2  0.276194   0.041938   6.586 1.15e-10 ***
## tempr.l2 -0.079337   0.044679  -1.776  0.07639 .
## part.l2   0.068082   0.025286   2.692  0.00733 **
## const    56.098652   5.916618   9.482  < 2e-16 ***
## trend    -0.011042   0.001992  -5.543 4.84e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 5.295 on 498 degrees of freedom
## Multiple R-Squared: 0.7227,  Adjusted R-squared: 0.7188
## F-statistic: 185.4 on 7 and 498 DF,  p-value: < 2.2e-16
```

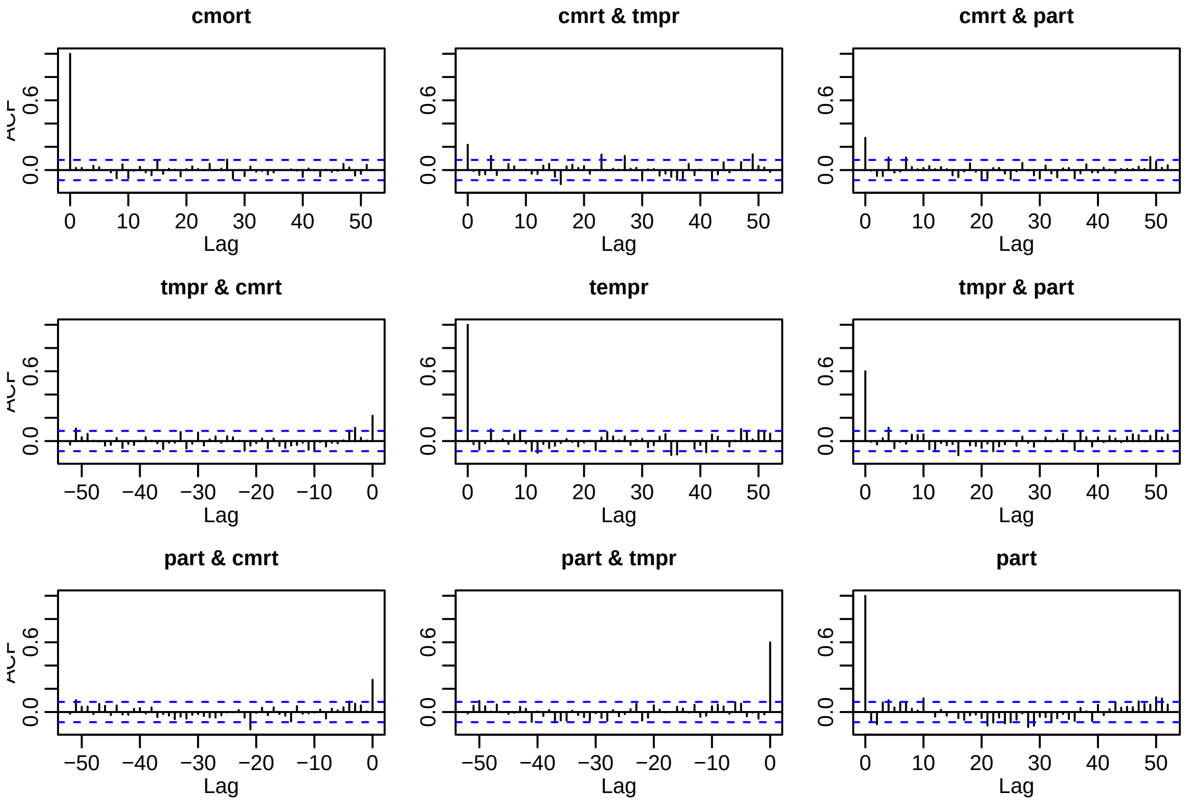
```
##
##
## Estimation results for equation tempr:
## ======================================
## tempr = cmort.l1 + tempr.l1 + part.l1 + cmort.l2 + tempr.l2 + part.l2 + const + trend
##
##            Estimate Std. Error t value Pr(>|t|)
## cmort.l1 -0.108889   0.050667  -2.149  0.03211 *
## tempr.l1  0.260963   0.051292   5.088 5.14e-07 ***
## part.l1  -0.050542   0.027844  -1.815  0.07010 .
## cmort.l2 -0.040870   0.048587  -0.841  0.40065
## tempr.l2  0.355592   0.051762   6.870 1.93e-11 ***
## part.l2  -0.095114   0.029295  -3.247  0.00125 **
## const    49.880485   6.854540   7.277 1.34e-12 ***
## trend    -0.004754   0.002308  -2.060  0.03993 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 6.134 on 498 degrees of freedom
## Multiple R-Squared: 0.5445,  Adjusted R-squared: 0.5381
## F-statistic: 85.04 on 7 and 498 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation part:
## =====================================
## part = cmort.l1 + tempr.l1 + part.l1 + cmort.l2 + tempr.l2 + part.l2 + const + trend
##
##            Estimate Std. Error t value Pr(>|t|)
## cmort.l1  0.078934   0.091773   0.860 0.390153
## tempr.l1 -0.388808   0.092906  -4.185 3.37e-05 ***
## part.l1   0.388814   0.050433   7.709 6.92e-14 ***
## cmort.l2 -0.325112   0.088005  -3.694 0.000245 ***
## tempr.l2  0.052780   0.093756   0.563 0.573724
## part.l2   0.382193   0.053062   7.203 2.19e-12 ***
## const    59.586169  12.415669   4.799 2.11e-06 ***
## trend    -0.007582   0.004180  -1.814 0.070328 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
##
## Residual standard error: 11.11 on 498 degrees of freedom
## Multiple R-Squared: 0.4679,  Adjusted R-squared: 0.4604
## F-statistic: 62.57 on 7 and 498 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##        cmort  tempr   part
## cmort 28.034  7.076  16.33
## tempr  7.076 37.627  40.88
## part  16.325 40.880 123.45
##
## Correlation matrix of residuals:
##        cmort  tempr   part
## cmort 1.0000 0.2179 0.2775
## tempr 0.2179 1.0000 0.5998
## part  0.2775 0.5998 1.0000
```

$\hat{M}_t = 56 - .01t + .3M_{t-1} - .2T_{t-1} + .04P_{t-1} + .28M_{t-2} - .08T_{t-2} + .07P_{t-2}$

To examine the residuals, we can plot the cross-correlations of the resid- uals and examine the multivariate version of the Q-test as follows:

```
acf(resid(fit), 52)
```

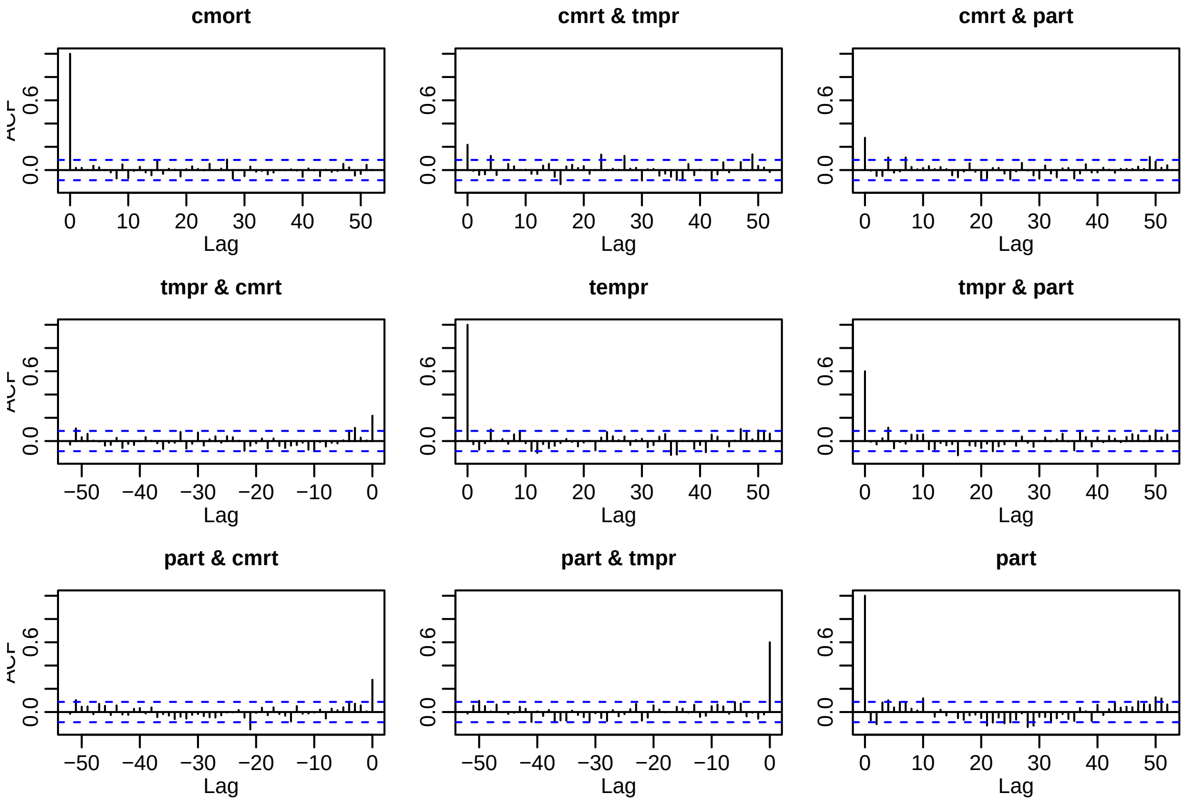


ACFs (diagonals) and CCFs (off-diagonals) for the residuals of the three-dimensional VAR(2). On the off-diagonals, the second-named series is the one that leads.

If the title of the off-diagonal plot is `x & y`, then y leads in the graphic; that is, on the upper-diagonal, the plot shows `corr[x(t+Lag), y(t)]` whereas in the lower-diagonal, if the title is x & y, you get a plot of `corr[x(t+Lag), y(t)]` (yes, it is the same thing, but the lags are negative in the lower diagonal). The graphic is labeled in a strange way, just remember the second named series is the one that leads.

We notice that most of the correlations in the residual series are negligible, however, the zero-order correlations of mortality with tempera- ture residuals is about .22 and mortality with particulate residuals is about .28

```
acf(resid(fit), 52)$acf
```




```
## , , 1
##
##                 [,1]         [,2]         [,3]
##  [1,]  1.0000000000  0.217879438  0.277509279
##  [2,]  0.0190479748  0.005389606  0.005820489
##  [3,]  0.0196665705  0.029266506  0.057572004
##  [4,]  0.0018750976  0.113509001  0.072168281
##  [5,]  0.0370667228  0.089215447  0.089481560
##  [6,]  0.0234497703  0.006534372  0.041550540
##  [7,]  0.0002439599 -0.020245959  0.015288896
##  [8,] -0.0201621665 -0.014605120  0.028696187
##  [9,] -0.0717493834 -0.047291658 -0.055464437
## [10,]  0.0488793637 -0.008198369  0.021221599
## [11,] -0.0695031498 -0.080878807 -0.004731138
## [12,] -0.0070995080 -0.074345678 -0.011980420
## [13,]  0.0282316571 -0.014114181 -0.014007958
## [14,] -0.0198756141 -0.034144315  0.051026255
## [15,] -0.0443005551 -0.038874943 -0.081377952
## [16,]  0.0786827878 -0.059140216 -0.031402333
## [17,] -0.0327084309 -0.041759473 -0.016714248
## [18,]  0.0108442536  0.022255941  0.040101939
## [19,]  0.0043285647 -0.062265045 -0.027016343
```

```
## [20,] -0.0554667399  0.022056093  0.037211470
## [21,]  0.0079066206 -0.018238253 -0.007438298
## [22,]  0.0314774049 -0.040331764 -0.149078216
## [23,]  0.0096106673 -0.082998640 -0.047516759
## [24,] -0.0025419428  0.003317043  0.017429386
## [25,]  0.0539863170  0.032757468 -0.003150329
## [26,] -0.0005172218  0.041461399 -0.004494827
## [27,]  0.0127625494 -0.014332058 -0.028122241
## [28,]  0.0904734843  0.040357750 -0.047025945
## [29,] -0.0746643793  0.015222129 -0.044827848
## [30,] -0.0008245447 -0.038980932 -0.034558792
## [31,] -0.0520697165  0.073548481 -0.017026157
## [32,]  0.0304587464 -0.022986878 -0.024226714
## [33,] -0.0141854337 -0.064576026 -0.056280760
## [34,] -0.0070769444  0.077901477 -0.040394100
## [35,] -0.0362246932 -0.014366382 -0.060734195
## [36,] -0.0214401093 -0.014338261 -0.030299110
## [37,] -0.0008471980 -0.068583531 -0.017670940
## [38,] -0.0006026269 -0.019236217 -0.043214529
## [39,]  0.0064111064  0.002799474  0.039626533
## [40,] -0.0043898669  0.034082897 -0.012423884
## [41,] -0.0583439171 -0.002440935  0.034561440
## [42,]  0.0136243581 -0.033637019  0.027505399
## [43,] -0.0033203555 -0.024584074 -0.024374319
## [44,] -0.0537574770 -0.059025968 -0.021135711
## [45,]  0.0030748770  0.027478454  0.056947794
## [46,] -0.0152810216 -0.032749931 -0.025585528
## [47,] -0.0075957603 -0.039362742  0.053481937
## [48,]  0.0532566114 -0.004024262  0.070240327
## [49,]  0.0235818271  0.003943218 -0.018320176
## [50,] -0.0474990754  0.062016271  0.047488310
## [51,] -0.0334291892  0.032924414  0.045649735
## [52,]  0.0451385770  0.108830728  0.103113081
## [53,]  0.0016785953 -0.030510828 -0.015254719
##
## , , 2
##
##                [,1]          [,2]          [,3]
##  [1,]  0.2178794381  1.0000000000  5.998217e-01
##  [2,] -0.0069134627 -0.0273153464 -2.028058e-02
```

```
##  [3,] -0.0417396384 -0.0732641900 -5.746080e-02
##  [4,] -0.0365251777 -0.0191031871 -5.234533e-03
##  [5,]  0.1223110978  0.0997388030 -3.702000e-02
##  [6,] -0.0426267519  0.0013077690  7.279985e-02
##  [7,]  0.0008907663  0.0168402620  8.611434e-02
##  [8,]  0.0527425268 -0.0236969942 -1.401886e-02
##  [9,]  0.0329738376  0.0578341857  4.853311e-02
## [10,]  0.0008613065  0.0881269759  6.528484e-02
## [11,]  0.0038643703 -0.0193190571  5.039434e-02
## [12,] -0.0322386983 -0.0849282312 -3.180028e-02
## [13,] -0.0337960274 -0.1018435407 -4.181919e-02
## [14,]  0.0383478236 -0.0263751966  6.256511e-02
## [15,]  0.0530821002 -0.0604857546 -4.390368e-03
## [16,] -0.0578959149 -0.0405959048  2.986441e-02
## [17,] -0.1207562981 -0.0172022414  4.496452e-02
## [18,]  0.0309632879  0.0176564288 -1.277802e-05
## [19,]  0.0458044264 -0.0082539630 -2.952468e-03
## [20,]  0.0181248335 -0.0458729285  2.196671e-02
## [21,]  0.0341990752 -0.0128123086  5.801552e-02
## [22,] -0.0318799968  0.0017036440 -4.847752e-02
## [23,]  0.0015451698 -0.0791214645 -7.364551e-02
## [24,]  0.1330538448  0.0298501082  6.925756e-02
## [25,]  0.0009457457  0.0770053571  2.507865e-02
## [26,]  0.0109719963  0.0405010344 -1.796386e-02
## [27,]  0.0031538570  0.0078040045 -3.669959e-02
## [28,]  0.1218428831  0.0390931973  1.646810e-02
## [29,]  0.0119989667 -0.0337334222 -7.647781e-02
## [30,]  0.0193617959  0.0084917093 -5.090994e-02
## [31,] -0.0902243555  0.0195814082 -1.051971e-02
## [32,]  0.0073009931 -0.0535270412 -7.326326e-02
## [33,]  0.0096090917 -0.0350304718 -4.321136e-02
## [34,] -0.0481614893  0.0380136660 -2.507569e-02
## [35,] -0.0305350325  0.0607175363  1.222326e-02
## [36,] -0.0572740455 -0.1200585756 -7.296693e-02
## [37,] -0.0847362741 -0.1160121279 -7.261332e-02
## [38,] -0.0768652106  0.0017602589 -8.082381e-02
## [39,]  0.0523430919  0.0006355169  1.791517e-02
## [40,] -0.0442097337 -0.0676960291 -3.356747e-02
## [41,]  0.0032764001 -0.0343545757  5.942883e-03
## [42,] -0.0015313453 -0.0959785808 -8.464231e-02
```

```
## [43,] -0.0724475785  0.0561233233  2.837262e-02
## [44,] -0.0374797220  0.0378247513  4.592497e-02
## [45,]  0.0672385341 -0.0004920405 -6.144555e-03
## [46,] -0.0198744796 -0.0468162051 -1.651516e-02
## [47,] -0.0003644189  0.0043254720 -2.643725e-03
## [48,]  0.0696476317  0.1028146531  6.472267e-02
## [49,] -0.0011331338  0.0738081817  5.243119e-03
## [50,]  0.1348519820  0.0142999698  5.172204e-02
## [51,]  0.0356162974  0.0924132277  9.607002e-02
## [52,]  0.0219433194  0.0778897000  5.430037e-02
## [53,] -0.0167431902  0.0649301370 -1.286292e-02
##
## , , 3
##
##              [,1]          [,2]          [,3]
##  [1,]  0.277509279  5.998217e-01  1.0000000000
##  [2,] -0.005606983 -6.272803e-03 -0.0718869340
##  [3,] -0.050471197 -2.927774e-02 -0.1044093310
##  [4,] -0.051476471  2.435578e-02  0.0818592172
##  [5,]  0.108826391  1.151831e-01  0.1014317196
##  [6,] -0.022393526 -6.285620e-02  0.0399776248
##  [7,] -0.011671622 -8.293098e-03  0.0868554043
##  [8,]  0.107868824 -2.285020e-02  0.0788537879
##  [9,]  0.026654486  5.549410e-02  0.0286138179
## [10,]  0.006512363  5.259021e-02  0.0134036524
## [11,]  0.017700351  5.772317e-02  0.1179402578
## [12,]  0.034172710 -7.100005e-02 -0.0005368955
## [13,]  0.008708124 -7.024868e-02 -0.0402356322
## [14,]  0.026161657 -1.668984e-02  0.0198641366
## [15,]  0.008635961 -3.657212e-02 -0.0296797994
## [16,] -0.045548433 -2.637129e-02 -0.0014683183
## [17,] -0.061769793 -1.222882e-01 -0.0526359004
## [18,] -0.012604978  4.833238e-05 -0.0665361808
## [19,]  0.058612968 -3.741204e-02 -0.0279237208
## [20,] -0.014398202 -4.194695e-02 -0.0219147857
## [21,] -0.077601777 -5.893514e-02 -0.0523712541
## [22,] -0.070870854 -2.192678e-02 -0.1158347237
## [23,]  0.017674360 -8.745858e-02 -0.0902506776
## [24,]  0.018570603 -4.593930e-02 -0.0473447030
## [25,] -0.030733137 -2.834240e-02 -0.0974021202
```

```
## [26,] -0.079026054  1.409404e-03 -0.0893268387
## [27,] -0.011309736 -4.065380e-02 -0.0655419435
## [28,]  0.061413962  3.922000e-02 -0.0118739795
## [29,] -0.004480642 -1.555248e-02 -0.1293403808
## [30,] -0.046126569 -4.969208e-02 -0.1167476498
## [31,] -0.074081955  6.289827e-04 -0.0410670937
## [32,]  0.039199340  3.136864e-02 -0.0436366189
## [33,] -0.029864342 -3.633935e-03 -0.0893552371
## [34,] -0.062001622  1.449538e-02 -0.0535228109
## [35,]  0.014113252  6.231999e-02 -0.0164022658
## [36,]  0.018152519 -3.877261e-03 -0.0598618895
## [37,] -0.073200277 -7.932530e-02 -0.0767545504
## [38,] -0.029641931  7.458741e-02  0.0355013579
## [39,]  0.049829939  3.407308e-02  0.0065261069
## [40,] -0.020708624 -4.724998e-02 -0.0758863362
## [41,] -0.020351186  3.246480e-02  0.0624133451
## [42,]  0.020070135 -9.845861e-03 -0.0263576348
## [43,]  0.004443117  4.202227e-02  0.0249697264
## [44,] -0.023152376  2.120964e-02  0.0813184577
## [45,]  0.009773793 -9.803032e-03  0.0384603537
## [46,]  0.009824226  3.559850e-02  0.0455509389
## [47,]  0.009718289  5.872142e-02  0.0414817491
## [48,]  0.029567261  5.085948e-02  0.0930198800
## [49,]  0.010326438 -2.081853e-03  0.0687912111
## [50,]  0.112706488  4.545080e-02  0.0631379931
## [51,]  0.073618289  9.333858e-02  0.1271988308
## [52,]  0.019653903  3.141887e-02  0.1165514742
## [53,]  0.041510649  5.583284e-02  0.0648846998
```

This means that the AR model is not capturing the concurrent effect of temperature and pollution on mortality (recall the data evolves over a week)

Thus, not unexpectedly, the Q-test rejects the null hypothesis that the noise is white.

```
serial.test(fit, lags.pt = 12, type="PT.adjusted")
```

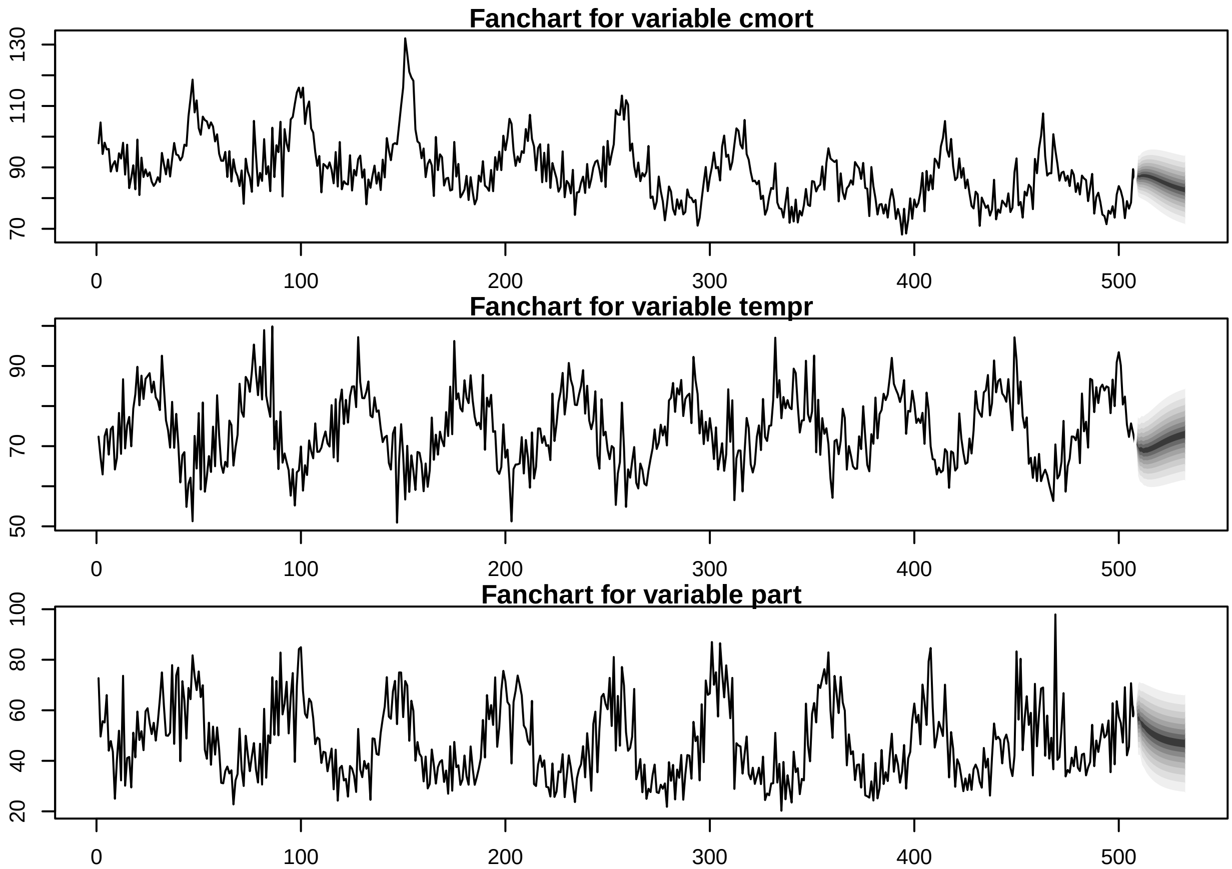
```
##
##  Portmanteau Test (adjusted)
##
## data:  Residuals of VAR object fit
## Chi-squared = 162.35, df = 90, p-value = 4.602e-06
```

```
# p-value is small, reject the null hypothesis
```

Now, we do predicitons.

Predictions from a VAR(2) fit to the LA mortality – pollution data.

```
fit.pr = predict(fit, n.ahead = 24, ci = 0.95) # 4 weeks ahead
par(mar=c(2,2,1,1))
fanchart(fit.pr) # plot prediction + error
```



We note that the package stripped time when plotting the fanchart and the horizontal axis is labeled 1, 2, 3, ...