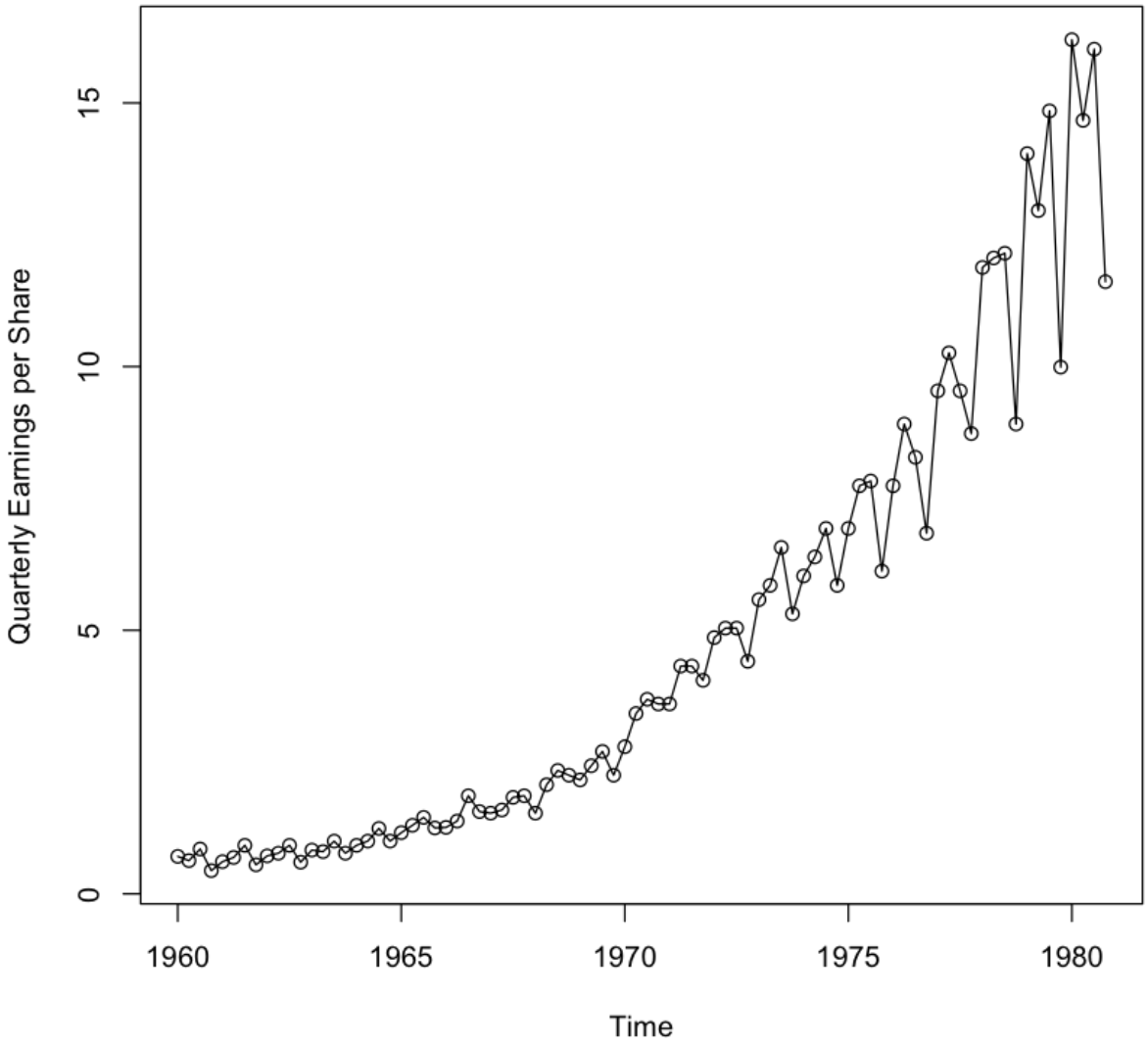
In [1]:
```
library(astsa)
```

In [2]:
```
plot(jj, type='o', ylab="Quarterly Earnings per Share")
```



In [3]:
```
decompose(jj)
```

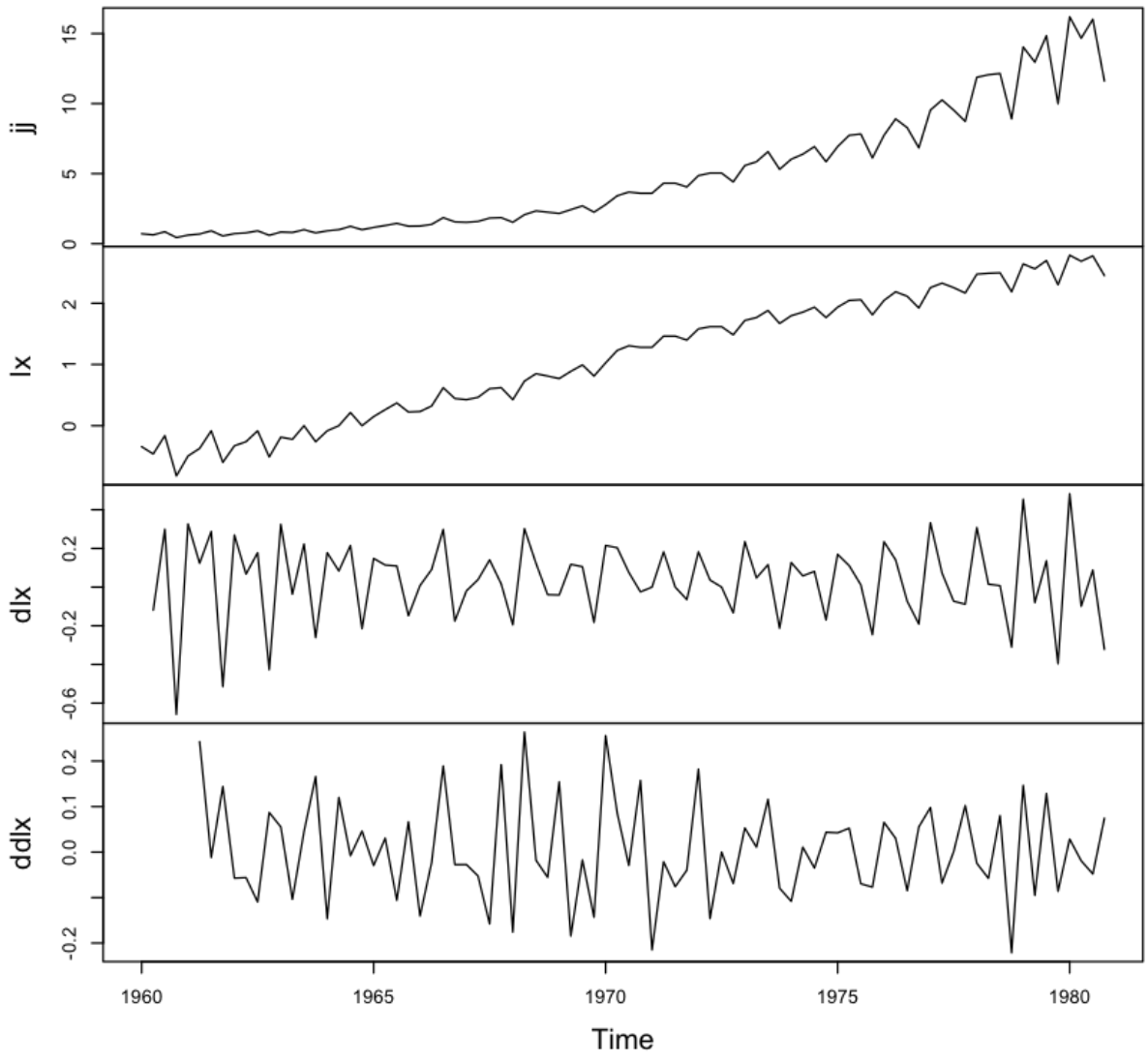
```
$x
          Qtr1      Qtr2      Qtr3      Qtr4
1960  0.710000  0.630000  0.850000  0.440000
1961  0.610000  0.690000  0.920000  0.550000
1962  0.720000  0.770000  0.920000  0.600000
1963  0.830000  0.800000  1.000000  0.770000
```

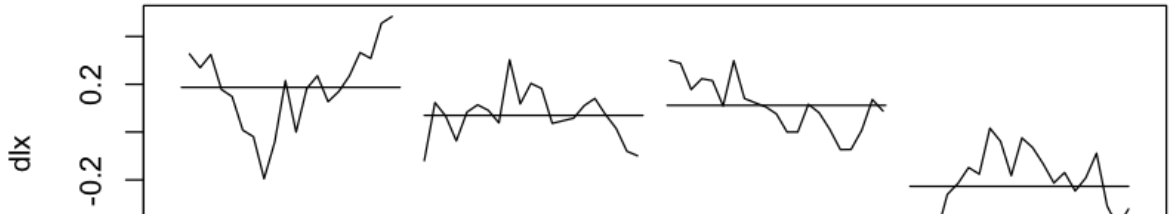
```
1964  0.920000  1.000000  1.240000  1.000000
1965  1.160000  1.300000  1.450000  1.250000
1966  1.260000  1.380000  1.860000  1.560000
1967  1.530000  1.590000  1.830000  1.860000
1968  1.530000  2.070000  2.340000  2.250000
1969  2.160000  2.430000  2.700000  2.250000
1970  2.790000  3.420000  3.690000  3.600000
1971  3.600000  4.320000  4.320000  4.050000
1972  4.860000  5.040000  5.040000  4.410000
1973  5.580000  5.850000  6.570000  5.310000
1974  6.030000  6.390000  6.930000  5.850000
1975  6.930000  7.740000  7.830000  6.120000
1976  7.740000  8.910000  8.280000  6.840000
1977  9.540000 10.260000  9.540000  8.729999
1978 11.880000 12.060000 12.150000  8.910000
1979 14.040000 12.960000 14.850000  9.990000
1980 16.200000 14.670000 16.020000 11.610000

$seasonal
           Qtr1       Qtr2       Qtr3       Qtr4
1960  0.2216094  0.2439844  0.3087344 -0.7743282
1961  0.2216094  0.2439844  0.3087344 -0.7743282
1962  0.2216094  0.2439844  0.3087344 -0.7743282
1963  0.2216094  0.2439844  0.3087344 -0.7743282
1964  0.2216094  0.2439844  0.3087344 -0.7743282
1965  0.2216094  0.2439844  0.3087344 -0.7743282
1966  0.2216094  0.2439844  0.3087344 -0.7743282
1967  0.2216094  0.2439844  0.3087344 -0.7743282
1968  0.2216094  0.2439844  0.3087344 -0.7743282
1969  0.2216094  0.2439844  0.3087344 -0.7743282
1970  0.2216094  0.2439844  0.3087344 -0.7743282
1971  0.2216094  0.2439844  0.3087344 -0.7743282
1972  0.2216094  0.2439844  0.3087344 -0.7743282
1973  0.2216094  0.2439844  0.3087344 -0.7743282
1974  0.2216094  0.2439844  0.3087344 -0.7743282
1975  0.2216094  0.2439844  0.3087344 -0.7743282
1976  0.2216094  0.2439844  0.3087344 -0.7743282
1977  0.2216094  0.2439844  0.3087344 -0.7743282
1978  0.2216094  0.2439844  0.3087344 -0.7743282
1979  0.2216094  0.2439844  0.3087344 -0.7743282
1980  0.2216094  0.2439844  0.3087344 -0.7743282

$trend
         Qtr1     Qtr2     Qtr3     Qtr4
1960       NA       NA  0.64500  0.64000
1961  0.65625  0.67875  0.70625  0.73000
1962  0.74000  0.74625  0.76625  0.78375
1963  0.79750  0.82875  0.86125  0.89750
1964  0.95250  1.01125  1.07000  1.13750
1965  1.20125  1.25875  1.30250  1.32500
```

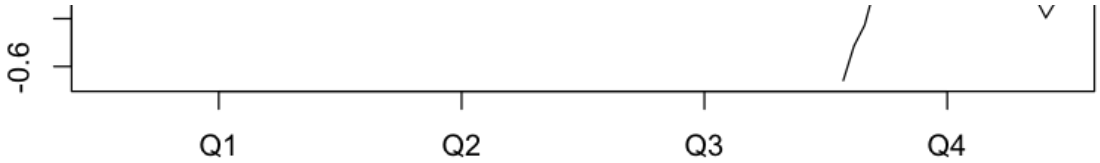
```
1966  1.38625  1.47625  1.54875  1.60875
1967  1.63125  1.66500  1.70250  1.76250
1968  1.88625  1.99875  2.12625  2.25000
1969  2.34000  2.38500  2.46375  2.66625
1970  2.91375  3.20625  3.47625  3.69000
1971  3.88125  4.01625  4.23000  4.47750
1972  4.65750  4.79250  4.92750  5.11875
1973  5.41125  5.71500  5.88375  6.00750
1974  6.12000  6.23250  6.41250  6.69375
1975  6.97500  7.12125  7.25625  7.50375
1976  7.70625  7.85250  8.16750  8.56125
1977  8.88750  9.28125  9.81000 10.32750
1978 10.87875 11.22750 11.52000 11.90250
1979 12.35250 12.82500 13.23000 13.71375
1980 14.07375 14.42250       NA       NA

$random
             Qtr1         Qtr2         Qtr3         Qtr4
1960           NA           NA -0.103734387  0.574328162
1961 -0.267859387 -0.232734388 -0.094984387  0.594328163
1962 -0.241609387 -0.220234388 -0.154984387  0.590578162
1963 -0.189109387 -0.272734388 -0.169984388  0.646828163
1964 -0.254109387 -0.255234388 -0.138734388  0.636828163
1965 -0.262859388 -0.202734388 -0.161234387  0.699328162
1966 -0.347859388 -0.340234388  0.002515613  0.725578163
1967 -0.322859388 -0.318984388 -0.181234387  0.871828163
1968 -0.577859387 -0.172734388 -0.094984388  0.774328162
1969 -0.401609388 -0.198984388 -0.072484387  0.358078163
1970 -0.345359387 -0.030234388 -0.094984387  0.684328163
1971 -0.502859388  0.059765612 -0.218734387  0.346828162
1972 -0.019109388  0.003515613 -0.196234388  0.065578163
1973 -0.052859387 -0.108984388  0.377515613  0.076828162
1974 -0.311609387 -0.086484388  0.208765613 -0.069421838
1975 -0.266609388  0.374765612  0.265015613 -0.609421838
1976 -0.187859387  0.813515613 -0.196234388 -0.946921837
1977  0.430890613  0.734765738 -0.578734137 -0.823172588
1978  0.779640863  0.588515738  0.321265613 -2.218171837
1979  1.465890613 -0.108984387  1.311265612 -2.949421838
1980  1.904640613  0.003515613           NA           NA

$figure
[1]  0.2216094  0.2439844  0.3087344 -0.7743282

$type
[1] "additive"

attr(,"class")
[1] "decomposed.ts"
```

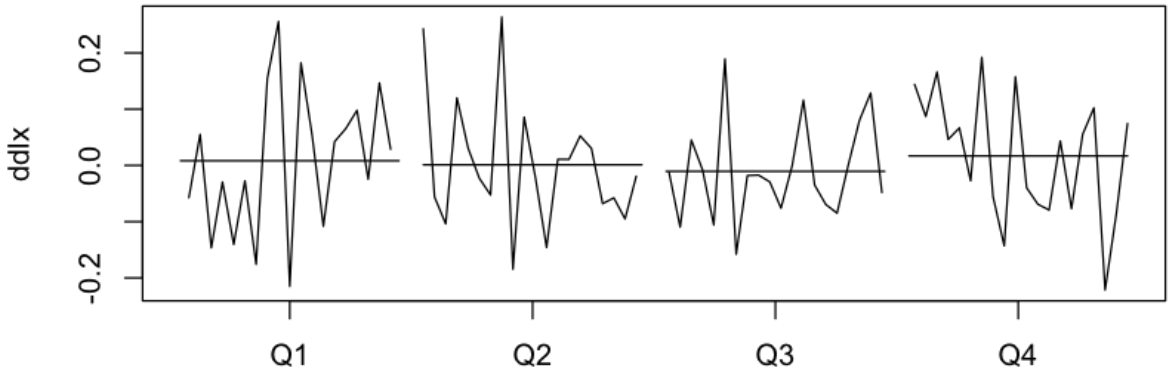
In [6]:
```
lx = log(jj)
```

```
dlx = diff(lx)
ddlx = diff(dlx, 4)
plot.ts(cbind(jj,lx, dlx, ddlx), main="")
par(mfrow=c(2,1))
monthplot(dlx); monthplot(ddlx)
```









In [7]:
```
acf2(ddlx, 50)
```

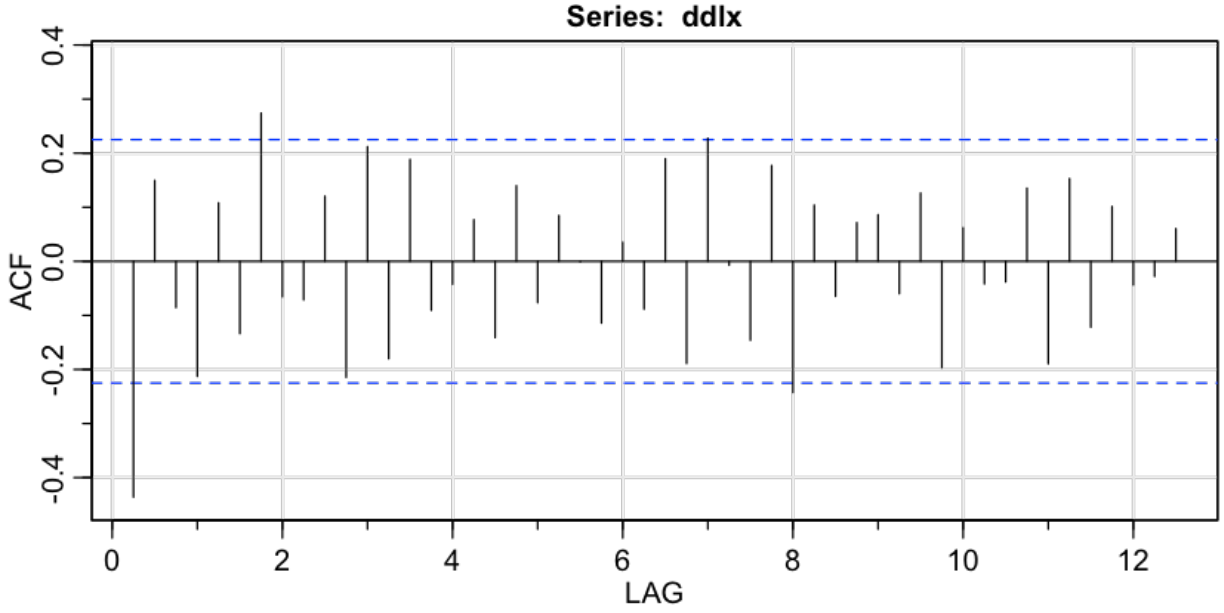
ACF	PACF
-0.44	-0.44
0.15	-0.05
-0.09	-0.05
-0.21	-0.33
0.11	-0.16
-0.13	-0.19
0.27	0.13
-0.07	0.08
-0.07	-0.14
0.12	0.04
-0.21	-0.03
0.21	0.14
-0.18	-0.06
0.19	0.07

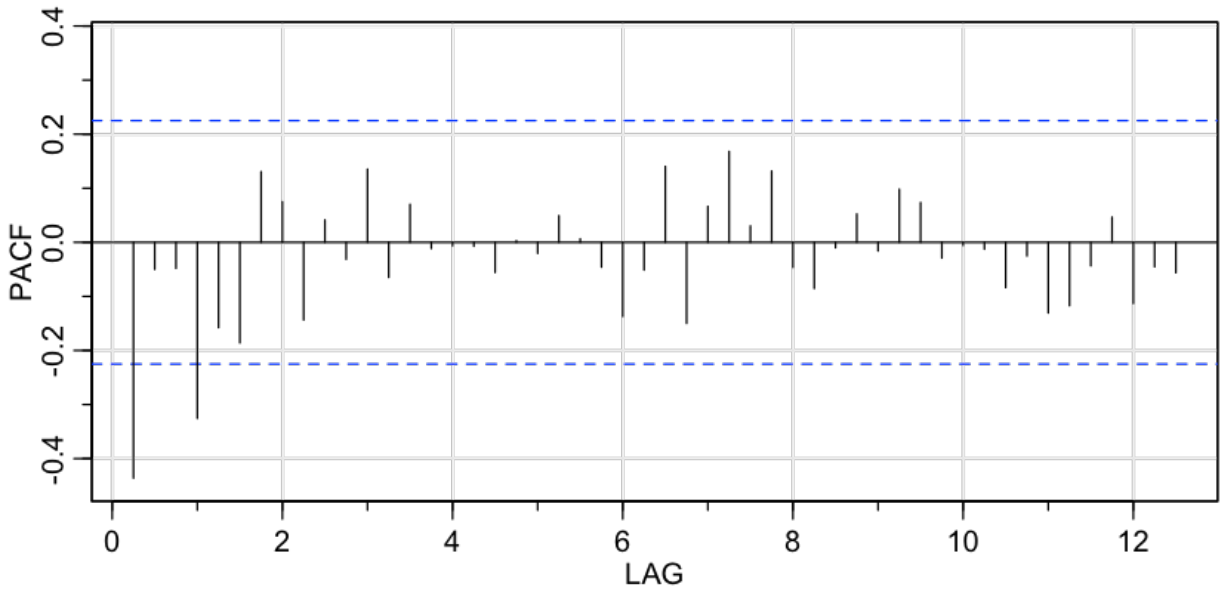
-0.09	-0.01
-0.04	-0.01
0.08	-0.01
-0.14	-0.06
0.14	0.00
-0.08	-0.02
0.08	0.05
0.00	0.01
-0.11	-0.05
0.04	-0.14
-0.09	-0.05
0.19	0.14
-0.19	-0.15
0.23	0.07
-0.01	0.17
-0.15	0.03
0.18	0.13
-0.24	-0.05
0.10	-0.09
-0.06	-0.01
0.07	0.05
0.09	-0.02
-0.06	0.10
0.13	0.07
-0.20	-0.03
0.06	-0.01
-0.04	-0.01
-0.04	-0.08
0.14	-0.03
-0.19	-0.13
0.15	-0.12
-0.12	-0.04
0.10	0.05

-0.04 -0.11

-0.03 -0.05

0.06 -0.06





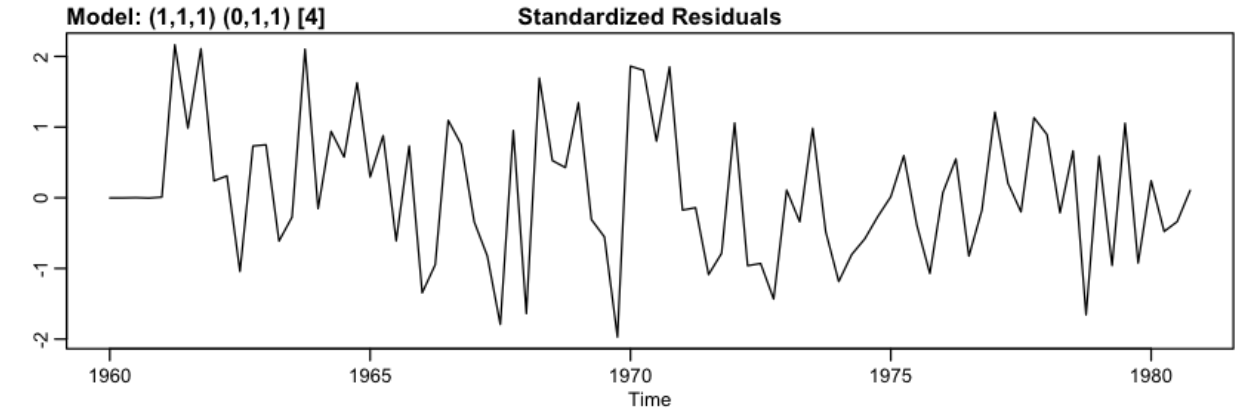
it appears that the acf is cutting off a lag 2s (s = 4) and pacf is tailing off at lags 1s, 2s..., these results imply that SMA(1), p=0, Q=1, s=4

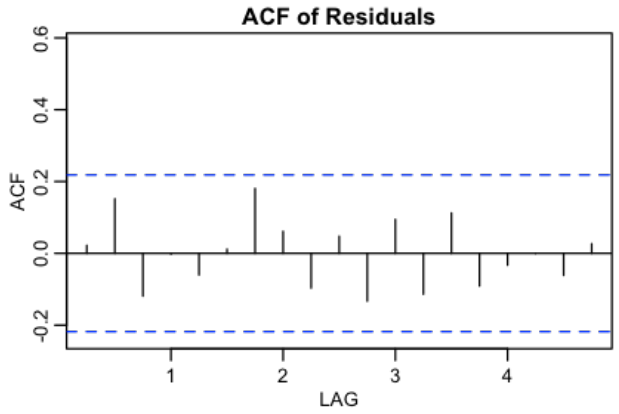
for the non seasonal component, we can see that the acf and pacf at the lower lags appear to tail off. So we have ARMA(1,1) with p=q=1

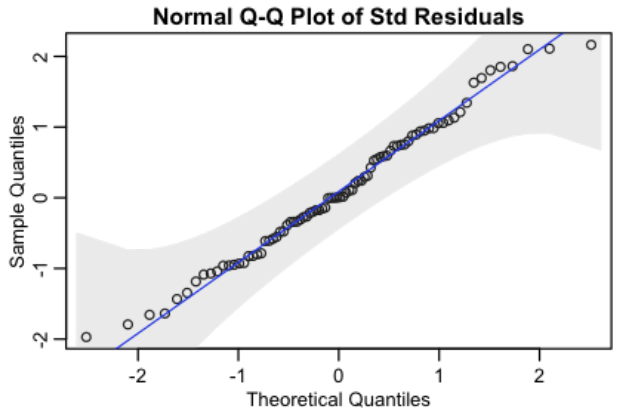
In [8]:
```
sarima(lx, 1, 1, 1, 0, 1,1,4)
```

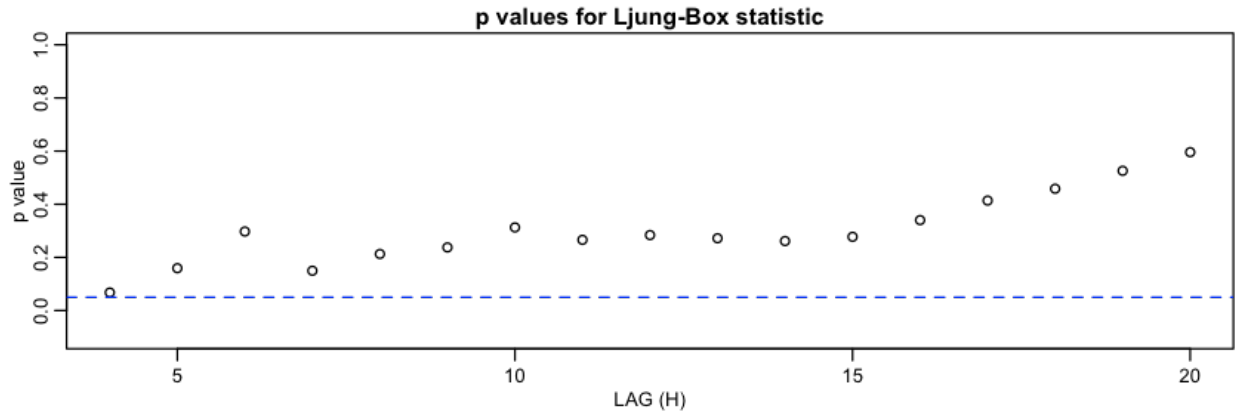
```
initial  value -2.242928
iter   2 value -2.375879
iter   3 value -2.430447
iter   4 value -2.438072
iter   5 value -2.456870
iter   6 value -2.465488
iter   7 value -2.465633
iter   8 value -2.465668
iter   9 value -2.465672
iter  10 value -2.465674
iter  11 value -2.465677
iter  12 value -2.465678
iter  13 value -2.465679
iter  14 value -2.465679
iter  14 value -2.465679
iter  14 value -2.465679
final  value -2.465679
converged
initial  value -2.402405
iter   2 value -2.406189
iter   3 value -2.408428
iter   4 value -2.410415
iter   5 value -2.411127
iter   6 value -2.411153
iter   7 value -2.411157
iter   8 value -2.411158
iter   8 value -2.411158
final  value -2.411158
converged

$fit

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c
(P, D,
    Q), period = S), include.mean = !no.constant, transform.pars = tr
ans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
         ar1      ma1     sma1
      0.0275  -0.6990  -0.3072
s.e.  0.2066   0.1646   0.1219

sigma^2 estimated as 0.007929:  log likelihood = 78.39,  aic = -148.7
7

$degrees_of_freedom
[1] 76
```

```
$ttable
     Estimate     SE t.value p.value
ar1    0.0275 0.2066  0.1331  0.8944
ma1   -0.6990 0.1646 -4.2476  0.0001
sma1  -0.3072 0.1219 -2.5203  0.0138

$AIC
[1] -1.814276

$AICc
[1] -1.810523

$BIC
[1] -1.698693
```







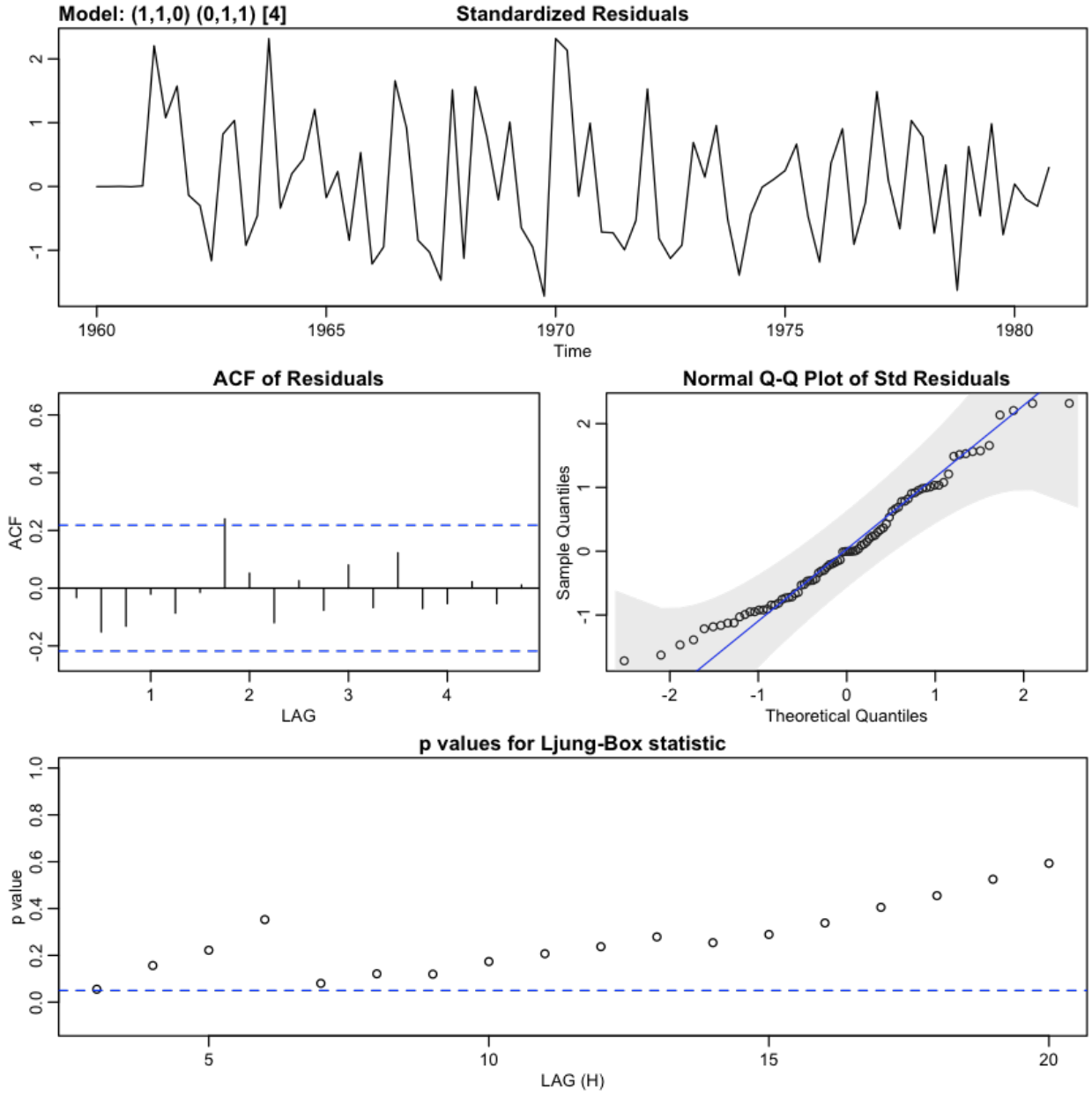


since the p value of the ar parameter doesn't look significant, we can drop the seaonsal component. So now we try ARIMA(0,1,1) x (0,1,1) and ARIMA(1,1,0) x (0,1,1)

In [9]:
```
sarima(lx,1,1,0,0,1,1,4)
```

```
initial  value -2.242928
iter   2 value -2.399148
iter   3 value -2.402522
iter   4 value -2.403053
iter   5 value -2.403056
iter   5 value -2.403056
iter   5 value -2.403056
final  value -2.403056
converged
initial  value -2.380113
iter   2 value -2.381008
iter   3 value -2.381084
iter   4 value -2.381084
iter   4 value -2.381084
iter   4 value -2.381084
final  value -2.381084
converged

$fit

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c
(P, D,
    Q), period = S), include.mean = !no.constant, transform.pars = tr
ans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
          ar1     sma1
      -0.5078  -0.3243
s.e.   0.1009   0.1046

sigma^2 estimated as 0.008472:  log likelihood = 76.01,  aic = -146.0
2

$degrees_of_freedom
[1] 77

$ttable
     Estimate     SE t.value p.value
ar1   -0.5078 0.1009 -5.0332  0.0000
sma1  -0.3243 0.1046 -3.1011  0.0027

$AIC
```

```
[1] -1.78072

$AICc
[1] -1.778868

$BIC
[1] -1.694033
```



Here we find that the p-value of the ar component is significant, so we use this model to forcaset the next 4 quarters

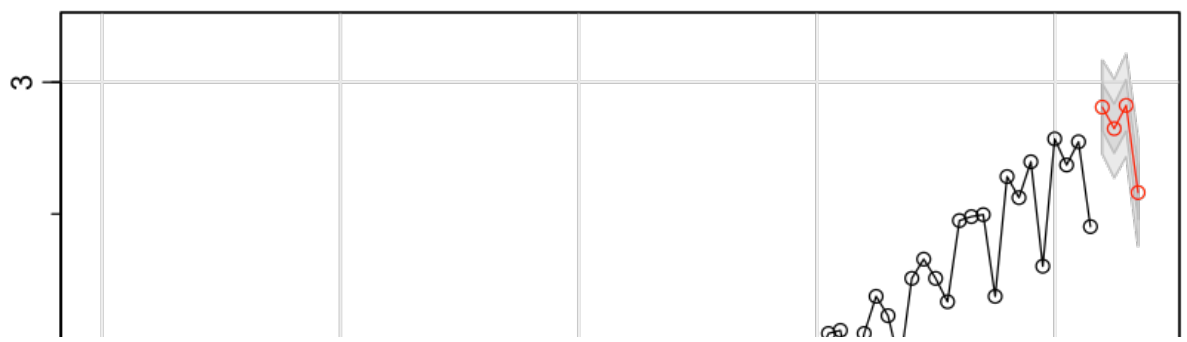
In [12]:
```
sarima.for(lx,4,0,1,1,0,1,1,4)
```

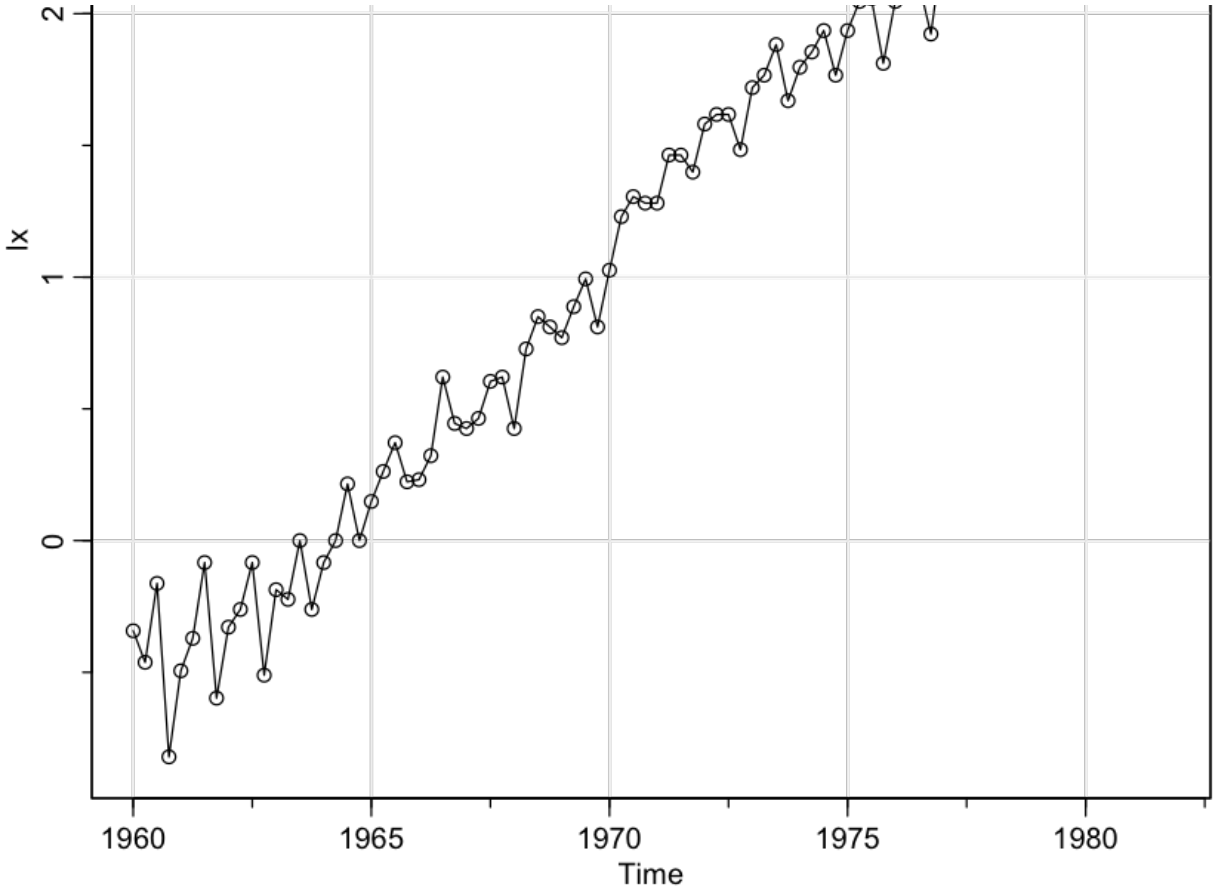
```
ERROR while rich displaying an object: Error in arr_partition(a, rows
```

```
, cols): rows >= 2L is not TRUE

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr
")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr
")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_html.list(obj)
9. repr_list_generic(obj, "html", "\t<li>%s</li>\n", "\t<dt>$%s</dt>\
n\t\t<dd>%s</dd>\n",
 .     "<strong>$%s</strong> = %s", "<ol>\n%s</ol>\n", "<dl>\n%s</dl>
\n",
 .     numeric_item = "\t<dt>[[%s]]</dt>\n\t\t<dd>%s</dd>\n", escape_
fun = html_escape)
10. lapply(vec, format2repr[[fmt]])
11. FUN(X[[i]], ...)
12. repr_html.ts(X[[i]], ...)
13. repr_ts_generic(obj, repr_html.matrix, ...)
14. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
15. repr_matrix_generic(obj, "<table>\n%s%s</table>\n", "<thead><tr>%
s</tr></thead>\n",
  .     "<th></th>", "<th scope=col>%s</th>", "<tbody>\n%s</tbody>\n"
,
  .     "\t<tr>%s</tr>\n", "<th scope=row>%s</th>", "<td>%s</td>",
  .     escape_fun = html_escape_vec, ...)
16. ellip_limit_arr(flatten(x), rows, cols)
17. arr_partition(a, rows, cols)
18. stopifnot(rows >= 2L, cols >= 2L)
ERROR while rich displaying an object: Error in repr_matrix_generic(o
bj, "\n%s%s\n", sprintf("|%%s\n|%s|\n", : formal argument "cols" matc
hed by multiple actual arguments
```

```
Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr
")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr
")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_markdown.list(obj)
9. repr_list_generic(obj, "markdown", "%s. %s\n", "$%s\n:   %s\n",
 .     "**$%s** = %s", "%s\n\n", numeric_item = "[[%s]]\n:   %s\n",
 .     item_uses_numbers = TRUE, escape_fun = html_escape)
10. lapply(vec, format2repr[[fmt]])
11. FUN(X[[i]], ...)
12. repr_markdown.ts(X[[i]], ...)
13. repr_ts_generic(obj, repr_markdown.matrix, ...)
14. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
ERROR while rich displaying an object: Error in arr_partition(a, rows
, cols): rows >= 2L is not TRUE

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr
")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
```

```
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr
")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_latex.list(obj)
9. repr_list_generic(obj, "latex", "\\item %s\n", "\\item[\\$%s] %s\n
",
 .     "\\textbf{\\$%s} = %s", enum_wrap = "\\begin{enumerate}\n%s\\e
nd{enumerate}\n",
 .     named_wrap = "\\begin{description}\n%s\\end{description}\n",
 .     numeric_item = "\\item[{[[%s]]}] %s\n", escape_fun = latex_esc
ape)
10. lapply(vec, format2repr[[fmt]])
11. FUN(X[[i]], ...)
12. repr_latex.ts(X[[i]], ...)
13. repr_ts_generic(obj, repr_latex.matrix, ..., colspec = colspec)
14. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
15. repr_matrix_generic(obj, sprintf("\\begin{tabular}{%s}\n%%s%%s\\e
nd{tabular}\n",
  .     cols), "%s\\\\\n\\hline\n", "  &", " %s &", "%s", "\t%s\\\\\n
",
  .     "%s &", " %s &", escape_fun = latex_escape_vec, ...)
16. ellip_limit_arr(flatten(x), rows, cols)
17. arr_partition(a, rows, cols)
18. stopifnot(rows >= 2L, cols >= 2L)

$pred
         Qtr1     Qtr2     Qtr3     Qtr4
1981 2.905343 2.823891 2.912148 2.581085

$se
           Qtr1       Qtr2       Qtr3       Qtr4
1981 0.08905414 0.09347899 0.09770366 0.10175307
```





Very COOL!

NOTE : I used the code and explanations from the air passenger example in the textbook. I left variable names the same because I didn't really see it necessary to change them.