153project

Jiyoon Clover Jeong and Jin Kweon

3/24/2018

EDA

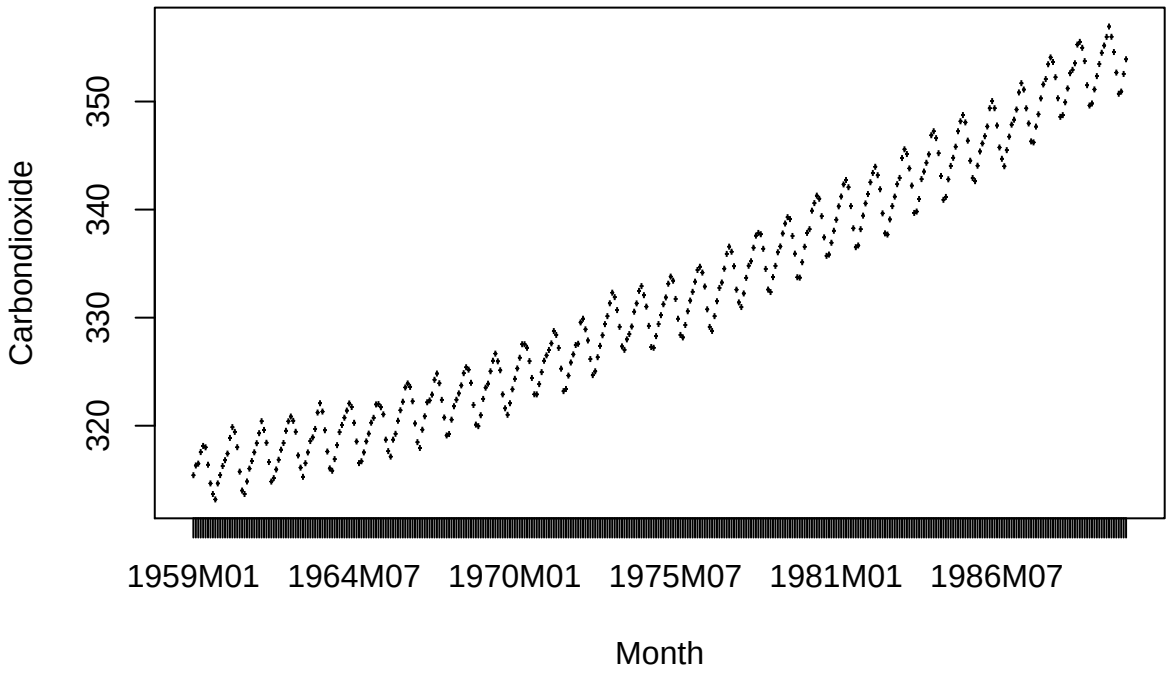
```
hawaii <- read.csv("../data/Carbon_Hawaii.csv")
dim(hawaii)
```

```
## [1] 384   2
```

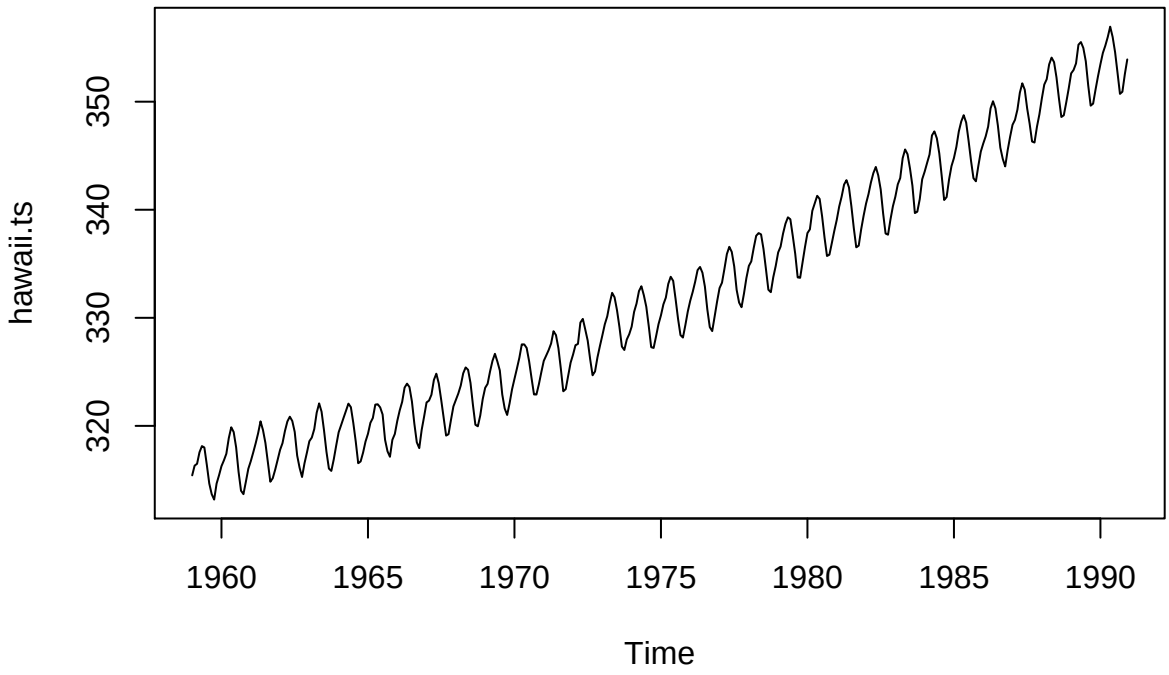
```
hawaii.ts <- ts(hawaii$Carbondioxide, frequency=12, start=c(1959, 1), end=c(1990, 12))
anyNA(hawaii.ts)
```

```
## [1] FALSE
```

```
plot(hawaii)
```



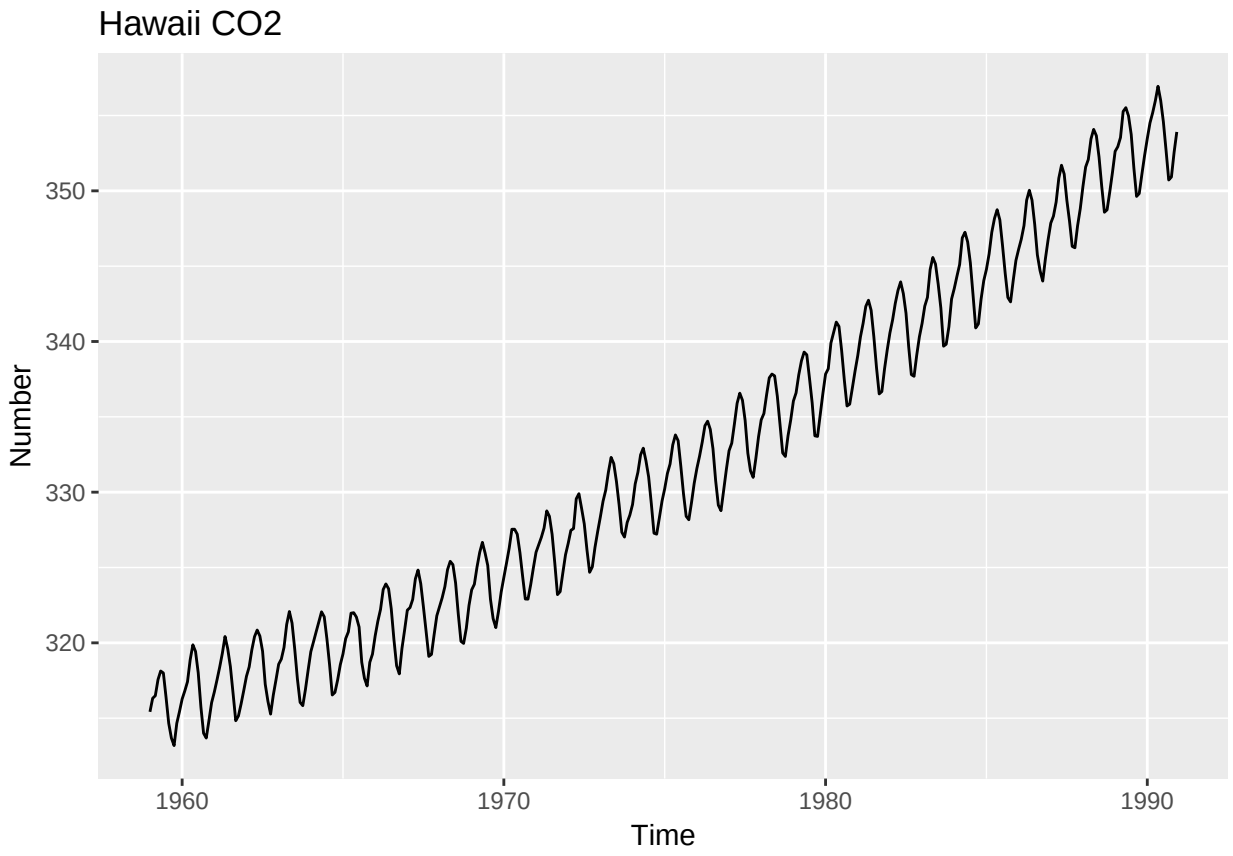

```
plot(hawaii.ts)
```



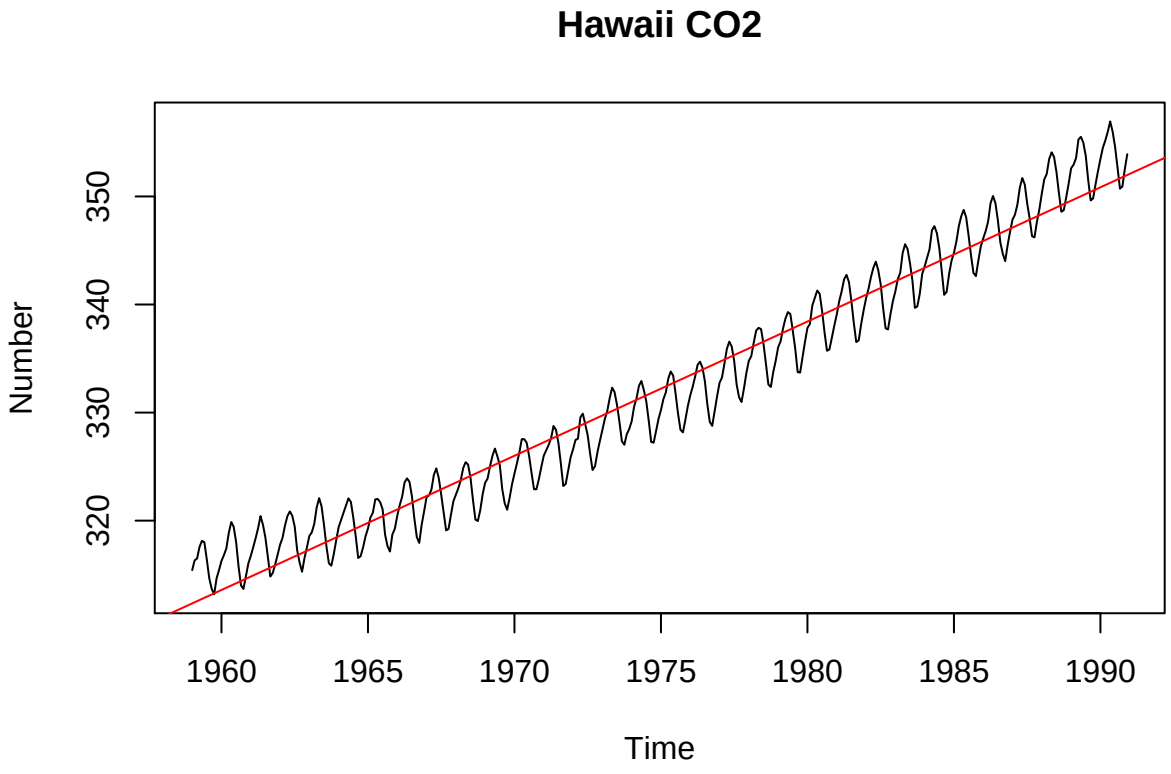

```
head(hawaii)
```

```
##     Month Carbondioxide
## 1 1959M01        315.42
## 2 1959M02        316.32
## 3 1959M03        316.49
## 4 1959M04        317.56
## 5 1959M05        318.13
## 6 1959M06        318.00
```

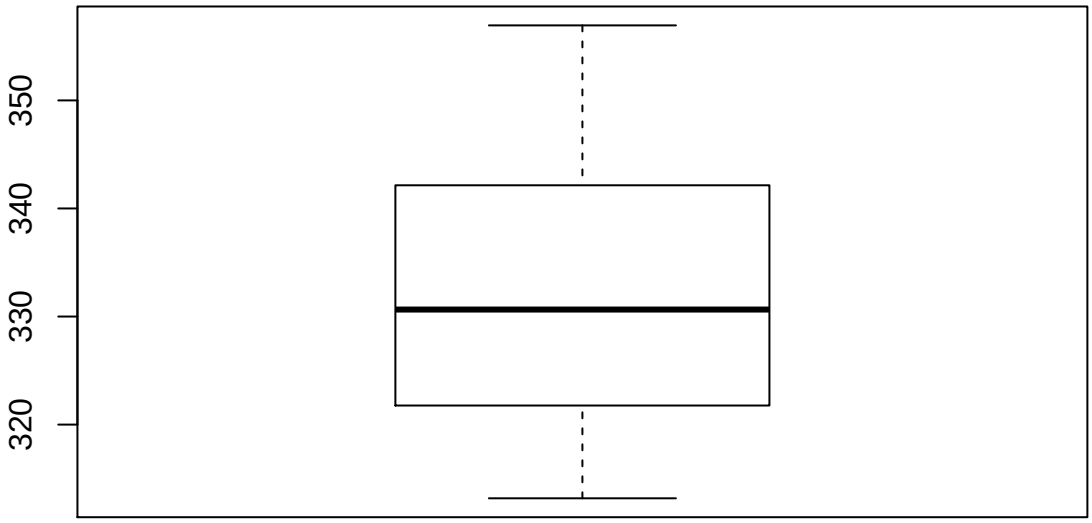
```
# Time series plot
autoplot(hawaii.ts, main = "Hawaii CO2", xlab = "Time", ylab = "Number")
```




```
plot.ts(hawaii.ts, main = "Hawaii CO2", xlab = "Time", ylab = "Number")
abline(reg=lm(hawaii.ts~time(hawaii.ts)), col = "red")
```



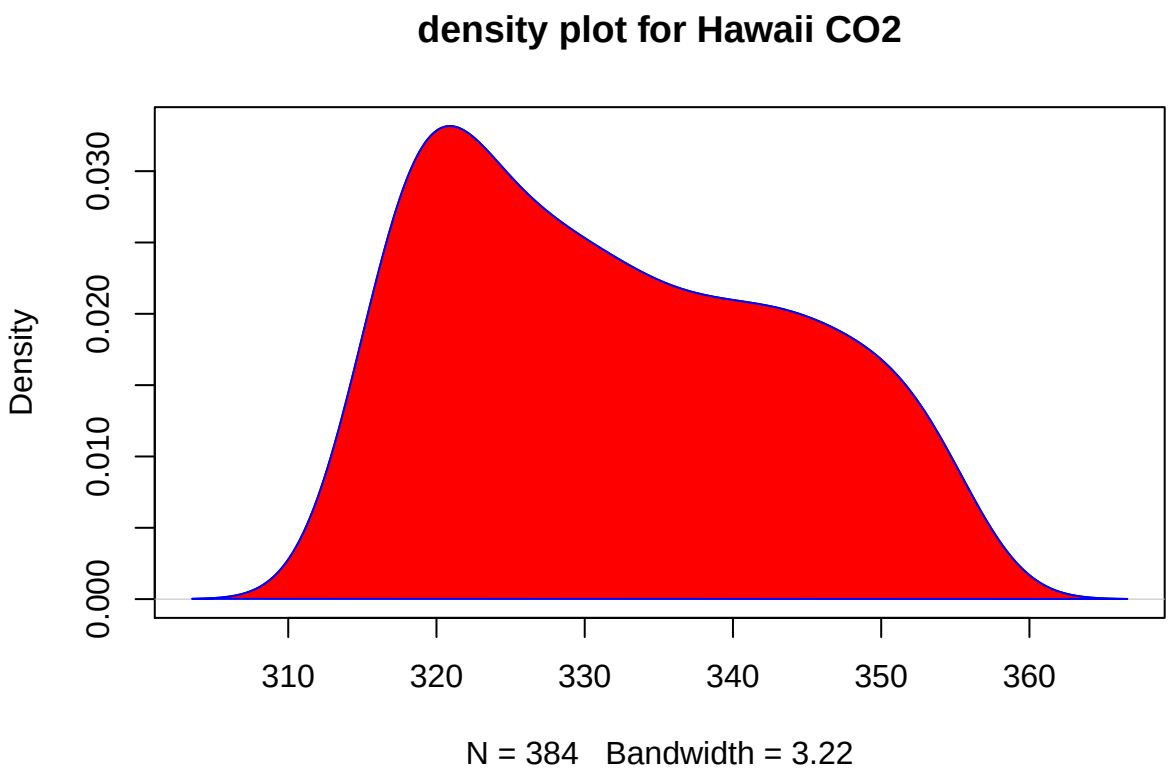
```r
# Identifying outliers
Out <- boxplot(hawaii.ts)$out
```




```r
length(which(hawaii.ts %in% Out))
```

```
## [1] 0
```

```r
den <- density(hawaii.ts, adjust = 1)
plot(den, main = "density plot for Hawaii CO2")
polygon(den, col = "red", border = "blue")
```




```r
outs <- tso(hawaii.ts, types = c("TC", "AO", "LS", "IO", "SLS"))
```

```
## Warning in locate.outliers.oloop(y = y, fit = fit, types = types, cval =
```

```
## cval, : the first 13 residuals were set to zero
```

```
tsoutliers(hawaii.ts)
```

```
## $index
## integer(0)
##
## $replacements
## numeric(0)
```

```
plot(outs)
```

```
## 'x' does not contain outliers to display
```

```
## NULL
```

```
# Another inspection
time(hawaii.ts)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 1959 1959.000 1959.083 1959.167 1959.250 1959.333 1959.417 1959.500
## 1960 1960.000 1960.083 1960.167 1960.250 1960.333 1960.417 1960.500
## 1961 1961.000 1961.083 1961.167 1961.250 1961.333 1961.417 1961.500
## 1962 1962.000 1962.083 1962.167 1962.250 1962.333 1962.417 1962.500
## 1963 1963.000 1963.083 1963.167 1963.250 1963.333 1963.417 1963.500
## 1964 1964.000 1964.083 1964.167 1964.250 1964.333 1964.417 1964.500
## 1965 1965.000 1965.083 1965.167 1965.250 1965.333 1965.417 1965.500
## 1966 1966.000 1966.083 1966.167 1966.250 1966.333 1966.417 1966.500
## 1967 1967.000 1967.083 1967.167 1967.250 1967.333 1967.417 1967.500
## 1968 1968.000 1968.083 1968.167 1968.250 1968.333 1968.417 1968.500
## 1969 1969.000 1969.083 1969.167 1969.250 1969.333 1969.417 1969.500
## 1970 1970.000 1970.083 1970.167 1970.250 1970.333 1970.417 1970.500
## 1971 1971.000 1971.083 1971.167 1971.250 1971.333 1971.417 1971.500
## 1972 1972.000 1972.083 1972.167 1972.250 1972.333 1972.417 1972.500
## 1973 1973.000 1973.083 1973.167 1973.250 1973.333 1973.417 1973.500
## 1974 1974.000 1974.083 1974.167 1974.250 1974.333 1974.417 1974.500
## 1975 1975.000 1975.083 1975.167 1975.250 1975.333 1975.417 1975.500
## 1976 1976.000 1976.083 1976.167 1976.250 1976.333 1976.417 1976.500
## 1977 1977.000 1977.083 1977.167 1977.250 1977.333 1977.417 1977.500
## 1978 1978.000 1978.083 1978.167 1978.250 1978.333 1978.417 1978.500
## 1979 1979.000 1979.083 1979.167 1979.250 1979.333 1979.417 1979.500
## 1980 1980.000 1980.083 1980.167 1980.250 1980.333 1980.417 1980.500
## 1981 1981.000 1981.083 1981.167 1981.250 1981.333 1981.417 1981.500
## 1982 1982.000 1982.083 1982.167 1982.250 1982.333 1982.417 1982.500
## 1983 1983.000 1983.083 1983.167 1983.250 1983.333 1983.417 1983.500
## 1984 1984.000 1984.083 1984.167 1984.250 1984.333 1984.417 1984.500
## 1985 1985.000 1985.083 1985.167 1985.250 1985.333 1985.417 1985.500
## 1986 1986.000 1986.083 1986.167 1986.250 1986.333 1986.417 1986.500
## 1987 1987.000 1987.083 1987.167 1987.250 1987.333 1987.417 1987.500
## 1988 1988.000 1988.083 1988.167 1988.250 1988.333 1988.417 1988.500
## 1989 1989.000 1989.083 1989.167 1989.250 1989.333 1989.417 1989.500
## 1990 1990.000 1990.083 1990.167 1990.250 1990.333 1990.417 1990.500
##           Aug      Sep      Oct      Nov      Dec
## 1959 1959.583 1959.667 1959.750 1959.833 1959.917
## 1960 1960.583 1960.667 1960.750 1960.833 1960.917
## 1961 1961.583 1961.667 1961.750 1961.833 1961.917
## 1962 1962.583 1962.667 1962.750 1962.833 1962.917
```

```
## 1963 1963.583 1963.667 1963.750 1963.833 1963.917
## 1964 1964.583 1964.667 1964.750 1964.833 1964.917
## 1965 1965.583 1965.667 1965.750 1965.833 1965.917
## 1966 1966.583 1966.667 1966.750 1966.833 1966.917
## 1967 1967.583 1967.667 1967.750 1967.833 1967.917
## 1968 1968.583 1968.667 1968.750 1968.833 1968.917
## 1969 1969.583 1969.667 1969.750 1969.833 1969.917
## 1970 1970.583 1970.667 1970.750 1970.833 1970.917
## 1971 1971.583 1971.667 1971.750 1971.833 1971.917
## 1972 1972.583 1972.667 1972.750 1972.833 1972.917
## 1973 1973.583 1973.667 1973.750 1973.833 1973.917
## 1974 1974.583 1974.667 1974.750 1974.833 1974.917
## 1975 1975.583 1975.667 1975.750 1975.833 1975.917
## 1976 1976.583 1976.667 1976.750 1976.833 1976.917
## 1977 1977.583 1977.667 1977.750 1977.833 1977.917
## 1978 1978.583 1978.667 1978.750 1978.833 1978.917
## 1979 1979.583 1979.667 1979.750 1979.833 1979.917
## 1980 1980.583 1980.667 1980.750 1980.833 1980.917
## 1981 1981.583 1981.667 1981.750 1981.833 1981.917
## 1982 1982.583 1982.667 1982.750 1982.833 1982.917
## 1983 1983.583 1983.667 1983.750 1983.833 1983.917
## 1984 1984.583 1984.667 1984.750 1984.833 1984.917
## 1985 1985.583 1985.667 1985.750 1985.833 1985.917
## 1986 1986.583 1986.667 1986.750 1986.833 1986.917
## 1987 1987.583 1987.667 1987.750 1987.833 1987.917
## 1988 1988.583 1988.667 1988.750 1988.833 1988.917
## 1989 1989.583 1989.667 1989.750 1989.833 1989.917
## 1990 1990.583 1990.667 1990.750 1990.833 1990.917
```

```
cycle(hawaii.ts)
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1959   1   2   3   4   5   6   7   8   9  10  11  12
## 1960   1   2   3   4   5   6   7   8   9  10  11  12
## 1961   1   2   3   4   5   6   7   8   9  10  11  12
## 1962   1   2   3   4   5   6   7   8   9  10  11  12
## 1963   1   2   3   4   5   6   7   8   9  10  11  12
## 1964   1   2   3   4   5   6   7   8   9  10  11  12
## 1965   1   2   3   4   5   6   7   8   9  10  11  12
## 1966   1   2   3   4   5   6   7   8   9  10  11  12
## 1967   1   2   3   4   5   6   7   8   9  10  11  12
## 1968   1   2   3   4   5   6   7   8   9  10  11  12
## 1969   1   2   3   4   5   6   7   8   9  10  11  12
## 1970   1   2   3   4   5   6   7   8   9  10  11  12
## 1971   1   2   3   4   5   6   7   8   9  10  11  12
## 1972   1   2   3   4   5   6   7   8   9  10  11  12
## 1973   1   2   3   4   5   6   7   8   9  10  11  12
## 1974   1   2   3   4   5   6   7   8   9  10  11  12
## 1975   1   2   3   4   5   6   7   8   9  10  11  12
## 1976   1   2   3   4   5   6   7   8   9  10  11  12
## 1977   1   2   3   4   5   6   7   8   9  10  11  12
## 1978   1   2   3   4   5   6   7   8   9  10  11  12
## 1979   1   2   3   4   5   6   7   8   9  10  11  12
## 1980   1   2   3   4   5   6   7   8   9  10  11  12
## 1981   1   2   3   4   5   6   7   8   9  10  11  12
```

```
## 1982   1   2   3   4   5   6   7   8   9  10  11  12
## 1983   1   2   3   4   5   6   7   8   9  10  11  12
## 1984   1   2   3   4   5   6   7   8   9  10  11  12
## 1985   1   2   3   4   5   6   7   8   9  10  11  12
## 1986   1   2   3   4   5   6   7   8   9  10  11  12
## 1987   1   2   3   4   5   6   7   8   9  10  11  12
## 1988   1   2   3   4   5   6   7   8   9  10  11  12
## 1989   1   2   3   4   5   6   7   8   9  10  11  12
## 1990   1   2   3   4   5   6   7   8   9  10  11  12
```

```
summary(hawaii.ts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   313.2   321.8   330.6   332.2   342.1   356.9
```

```
str(hawaii.ts)
```

```
##  Time-Series [1:384] from 1959 to 1991: 315 316 316 318 318 ...
```

```
dim(hawaii.ts)
```

```
## NULL
```

```
sd(hawaii.ts)
```

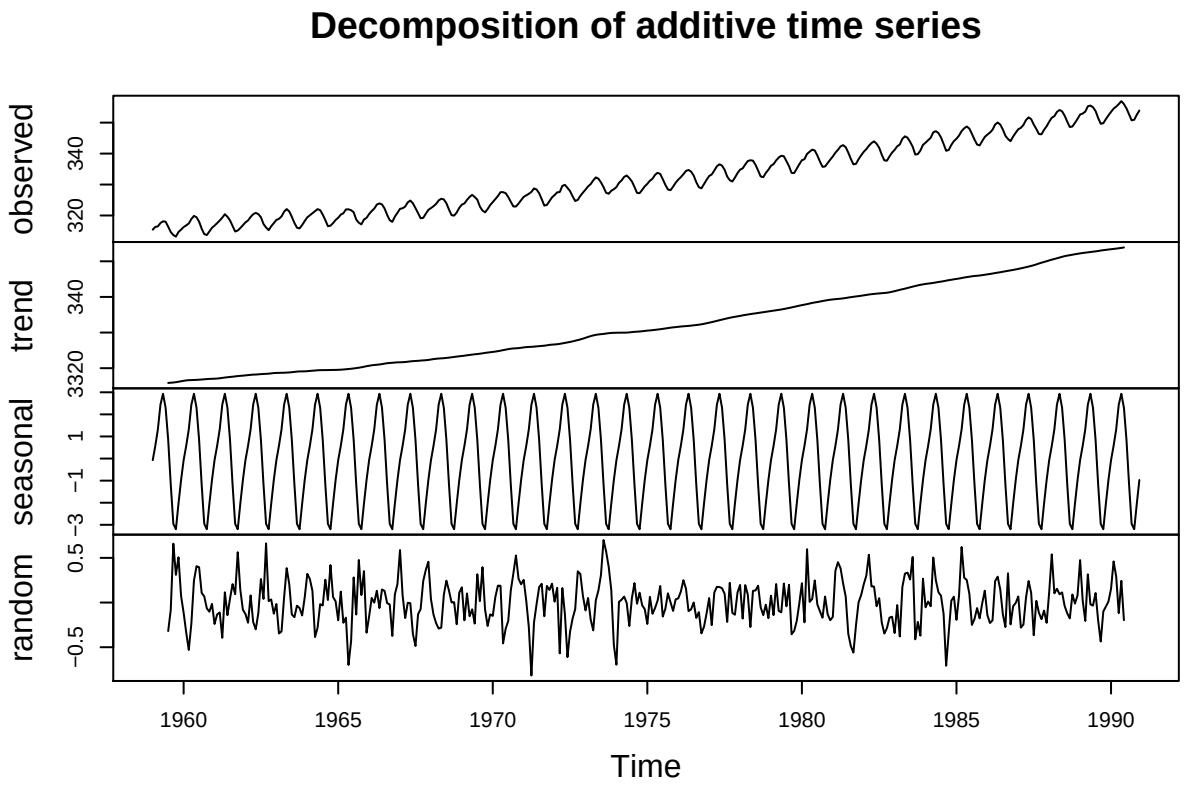
```
## [1] 11.76247
```

Analysis

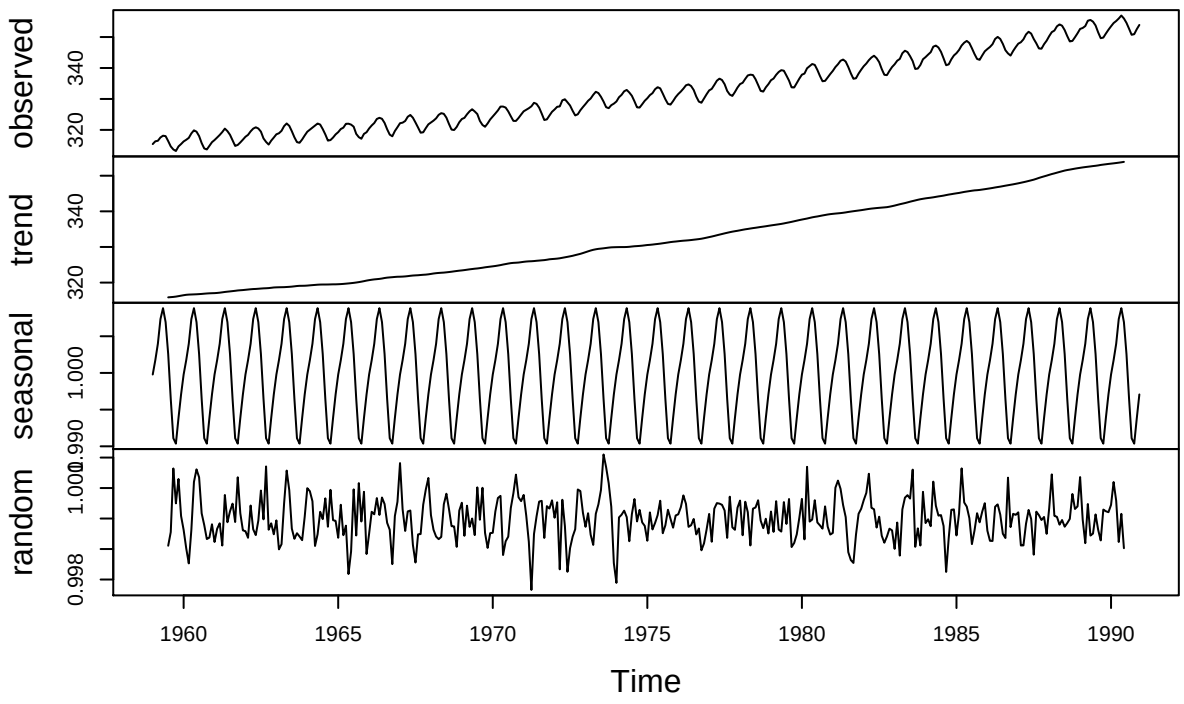
```
#Initial check
adf.test(hawaii.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  hawaii.ts
## Dickey-Fuller = -2.428, Lag order = 7, p-value = 0.3964
## alternative hypothesis: stationary
```

```
#kpss.test(hawaii.ts)




#Trend OR Seasonality
decomp <- decompose(hawaii.ts, type=c("additive")) # use type = "additive" for additive components by M
plot(decomp)
```




```
decomp2 <- decompose(hawaii.ts, type=c("multiplicative")) # use type = "additive" for additive componen
plot(decomp2)
```




```
lowesdecomp <- stl(hawaii.ts, s.window = "periodic") #seasonal decomposition by lowess
```

```
lowesdecomp
```

```
##  Call:
##  stl(x = hawaii.ts, s.window = "periodic")
##
## Components
##              seasonal    trend    remainder
## Jan 1959 -0.0801751 315.2364  0.263821819
## Feb 1959  0.5630906 315.3365  0.420459004
## Mar 1959  1.2747939 315.4365 -0.221341442
## Apr 1959  2.3934412 315.5362 -0.369648749
## May 1959  2.8908384 315.6359 -0.396705992
## Jun 1959  2.2901417 315.7309 -0.021042888
## Jul 1959  0.8516326 315.8259 -0.287567318
## Aug 1959 -1.1543733 315.9181 -0.103724055
## Sep 1959 -2.9275669 316.0103  0.597306823
## Oct 1959 -3.1649278 316.1178  0.227100765
## Nov 1959 -2.0010387 316.2254  0.435644767
## Dec 1959 -0.9358567 316.3316  0.034264694
## Jan 1960 -0.0801751 316.4378 -0.087614973
## Feb 1960  0.5630906 316.5180 -0.271113605
## Mar 1960  1.2747939 316.5983 -0.453049868
## Apr 1960  2.3934412 316.6492 -0.172650051
## May 1960  2.8908384 316.7002  0.278999829
## Jun 1960  2.2901417 316.7489  0.390944938
## Jul 1960  0.8516326 316.7977  0.360702512
## Aug 1960 -1.1543733 316.8466  0.057755333
## Sep 1960 -2.9275669 316.8956  0.031995769
## Oct 1960 -3.1649278 316.9317 -0.086745777
## Nov 1960 -2.0010387 316.9678 -0.126737262
## Dec 1960 -0.9358567 317.0018 -0.035975845
## Jan 1961 -0.0801751 317.0359 -0.225714021
## Feb 1961  0.5630906 317.0998 -0.122933961
## Mar 1961  1.2747939 317.1638 -0.058591531
## Apr 1961  2.3934412 317.2574 -0.340802283
## May 1961  2.8908384 317.3509  0.178237029
## Jun 1961  2.2901417 317.4462 -0.126361840
## Jul 1961  0.8516326 317.5415  0.026851757
## Aug 1961 -1.1543733 317.6280  0.166409705
## Sep 1961 -2.9275669 317.7144  0.043155269
## Oct 1961 -3.1649278 317.7860  0.528893704
## Nov 1961 -2.0010387 317.8577  0.093382199
## Dec 1961 -0.9358567 317.9177 -0.131826846
## Jan 1962 -0.0801751 317.9777 -0.117535484
## Feb 1962  0.5630906 318.0379 -0.201001346
## Mar 1962  1.2747939 318.0981  0.157095162
## Apr 1962  2.3934412 318.1583 -0.141761312
## May 1962  2.8908384 318.2185 -0.259367722
## Jun 1962  2.2901417 318.2777 -0.117860852
## Jul 1962  0.8516326 318.3369  0.251458483
## Aug 1962 -1.1543733 318.3879  0.016475980
## Sep 1962 -2.9275669 318.4389  0.608681091
## Oct 1962 -3.1649278 318.4917 -0.056743134
## Nov 1962 -2.0010387 318.5445 -0.013417300
```

```
## Dec 1962 -0.9358567 318.5920 -0.126096058
## Jan 1963 -0.0801751 318.6394  0.020725589
## Feb 1963  0.5630906 318.6707 -0.313789905
## Mar 1963  1.2747939 318.7019 -0.276743029
## Apr 1963  2.3934412 318.7361  0.090465101
## May 1963  2.8908384 318.7702  0.418923295
## Jun 1963  2.2901417 318.8261  0.193727132
## Jul 1963  0.8516326 318.8820 -0.153656565
## Aug 1963 -1.1543733 318.9456 -0.181256872
## Sep 1963 -2.9275669 319.0092 -0.031669564
## Oct 1963 -3.1649278 319.0554 -0.060497413
## Nov 1963 -2.0010387 319.1016 -0.190575202
## Dec 1963 -0.9358567 319.1419 -0.006053056
## Jan 1964 -0.0801751 319.1822  0.307969496
## Feb 1964  0.5630906 319.2364  0.270462268
## Mar 1964  1.2747939 319.2907  0.174517410
## Apr 1964  2.3934412 319.3411 -0.334525450
## May 1964  2.8908384 319.3915 -0.222318246
## Jun 1964  2.2901417 319.4214  0.018429535
## Jul 1964  0.8516326 319.4514 -0.033010218
## Aug 1964 -1.1543733 319.4695  0.224868011
## Sep 1964 -2.9275669 319.4876 -0.020066146
## Oct 1964 -3.1649278 319.5022  0.372685154
## Nov 1964 -2.0010387 319.5169  0.014186514
## Dec 1964 -0.9358567 319.5308 -0.044912942
## Jan 1965 -0.0801751 319.5447 -0.194511991
## Feb 1965  0.5630906 319.5779  0.139034011
## Mar 1965  1.2747939 319.6111 -0.155857617
## Apr 1965  2.3934412 319.6769 -0.100329681
## May 1965  2.8908384 319.7427 -0.633551682
## Jun 1965  2.2901417 319.8317 -0.411855370
## Jul 1965  0.8516326 319.9207  0.277653408
## Aug 1965 -1.1543733 320.0342 -0.169791572
## Sep 1965 -2.9275669 320.1476  0.429951062
## Oct 1965 -3.1649278 320.2813  0.023628996
## Nov 1965 -2.0010387 320.4150  0.296056989
## Dec 1965 -0.9358567 320.5460 -0.360151202
## Jan 1966 -0.0801751 320.6770 -0.136858986
## Feb 1966  0.5630906 320.7815  0.085381597
## Mar 1966  1.2747939 320.8860  0.059184550
## Apr 1966  2.3934412 320.9779  0.168656798
## May 1966  2.8908384 321.0698 -0.050620890
## Jun 1966  2.2901417 321.1696  0.130276732
## Jul 1966  0.8516326 321.2694  0.138986820
## Aug 1966 -1.1543733 321.3588  0.005557962
## Sep 1966 -2.9275669 321.4483 -0.040683281
## Oct 1966 -3.1649278 321.5146 -0.409720805
## Nov 1966 -2.0010387 321.5810  0.049991730
## Dec 1966 -0.9358567 321.6272  0.178688429
## Jan 1967 -0.0801751 321.6733  0.576885535
## Feb 1967  0.5630906 321.7169  0.059961311
## Mar 1967  1.2747939 321.7606 -0.155400543
## Apr 1967  2.3934412 321.8178  0.038724167
## May 1967  2.8908384 321.8751  0.064098941
```

```
## Jun 1967  2.2901417 321.9373 -0.297479902
## Jul 1967  0.8516326 321.9996 -0.461246280
## Aug 1967 -1.1543733 322.0634 -0.149010197
## Sep 1967 -2.9275669 322.1272 -0.099586500
## Oct 1967 -3.1649278 322.1994  0.195493240
## Nov 1967 -2.0010387 322.2717  0.289323040
## Dec 1967 -0.9358567 322.3581  0.377764068
## Jan 1968 -0.0801751 322.4445  0.035705502
## Feb 1968  0.5630906 322.5304 -0.103457845
## Mar 1968  1.2747939 322.6163 -0.161058823
## Apr 1968  2.3934412 322.6822 -0.215623579
## May 1968  2.8908384 322.7481 -0.228938272
## Jun 1968  2.2901417 322.8169  0.082920105
## Jul 1968  0.8516326 322.8858  0.232590947
## Aug 1968 -1.1543733 322.9745  0.099910805
## Sep 1968 -2.9275669 323.0631 -0.035581721
## Oct 1968 -3.1649278 323.1556 -0.030662012
## Nov 1968 -2.0010387 323.2480 -0.276992243
## Dec 1968 -0.9358567 323.3355  0.080307226
## Jan 1969 -0.0801751 323.4231  0.177107102
## Feb 1969  0.5630906 323.5183 -0.191415247
## Mar 1969  1.2747939 323.6136  0.151624773
## Apr 1969  2.3934412 323.7108 -0.094243626
## May 1969  2.8908384 323.8080 -0.028861960
## Jun 1969  2.2901417 323.8957 -0.225885615
## Jul 1969  0.8516326 323.9835  0.294903196
## Aug 1969 -1.1543733 324.0778 -0.023443194
## Sep 1969 -2.9275669 324.1722  0.365398030
## Oct 1969 -3.1649278 324.2729 -0.097991329
## Nov 1969 -2.0010387 324.3737 -0.292630630
## Dec 1969 -0.9358567 324.4687 -0.162869619
## Jan 1970 -0.0801751 324.5638 -0.143608203
## Feb 1970  0.5630906 324.6723  0.064643420
## Mar 1970  1.2747939 324.7807  0.234457413
## Apr 1970  2.3934412 324.9164  0.230159424
## May 1970  2.8908384 325.0521 -0.402888502
## Jun 1970  2.2901417 325.1893 -0.269453372
## Jul 1970  0.8516326 325.3266 -0.198205777
## Aug 1970 -1.1543733 325.4318  0.142581052
## Sep 1970 -2.9275669 325.5370  0.300555496
## Oct 1970 -3.1649278 325.6127  0.452246108
## Nov 1970 -2.0010387 325.6884  0.162686780
## Dec 1970 -0.9358567 325.7572  0.138674408
## Jan 1971 -0.0801751 325.8260  0.264162442
## Feb 1971  0.5630906 325.8802  0.066758785
## Mar 1971  1.2747939 325.9343 -0.199082502
## Apr 1971  2.3934412 325.9848 -0.758272436
## May 1971  2.8908384 326.0354 -0.166212306
## Jun 1971  2.2901417 326.1039  0.005990943
## Jul 1971  0.8516326 326.1724  0.176006658
## Aug 1971 -1.1543733 326.2581  0.176272049
## Sep 1971 -2.9275669 326.3438 -0.216274945
## Oct 1971 -3.1649278 326.4356  0.129312516
## Nov 1971 -2.0010387 326.5274  0.113650037
```

```
## Dec 1971 -0.9358567 326.6012  0.184623949
## Jan 1972 -0.0801751 326.6751  0.005098266
## Feb 1972  0.5630906 326.7545  0.152398857
## Mar 1972  1.2747939 326.8339 -0.528738184
## Apr 1972  2.3934412 326.9482  0.218403956
## May 1972  2.8908384 327.0624 -0.053203840
## Jun 1972  2.2901417 327.2079 -0.578011280
## Jul 1972  0.8516326 327.3534 -0.315006255
## Aug 1972 -1.1543733 327.5273 -0.202900584
## Sep 1972 -2.9275669 327.7012 -0.093607299
## Oct 1972 -3.1649278 327.8995  0.305454014
## Nov 1972 -2.0010387 328.0978  0.243265387
## Dec 1972 -0.9358567 328.3171  0.008770483
## Jan 1973 -0.0801751 328.5364 -0.086224015
## Feb 1973  0.5630906 328.7561  0.080842620
## Mar 1973  1.2747939 328.9757 -0.110528376
## Apr 1973  2.3934412 329.1577 -0.221146985
## May 1973  2.8908384 329.3397  0.079484469
## Jun 1973  2.2901417 329.4645  0.145373784
## Jul 1973  0.8516326 329.5893  0.259075565
## Aug 1973 -1.1543733 329.6763  0.628062330
## Sep 1973 -2.9275669 329.7633  0.504236711
## Oct 1973 -3.1649278 329.8210  0.363924452
## Nov 1973 -2.0010387 329.8787  0.112362253
## Dec 1973 -0.9358567 329.9018 -0.485954839
## Jan 1974 -0.0801751 329.9249 -0.664771525
## Feb 1974  0.5630906 329.9357  0.051216995
## Mar 1974  1.2747939 329.9464  0.098767885
## Apr 1974  2.3934412 329.9791  0.107459237
## May 1974  2.8908384 330.0118  0.017400653
## Jun 1974  2.2901417 330.0691 -0.279247358
## Jul 1974  0.8516326 330.1265  0.041917096
## Aug 1974 -1.1543733 330.1865  0.207878330
## Sep 1974 -2.9275669 330.2465 -0.038972821
## Oct 1974 -3.1649278 330.3097  0.065242538
## Nov 1974 -2.0010387 330.3728 -0.081792044
## Dec 1974 -0.9358567 330.4470 -0.101150107
## Jan 1975 -0.0801751 330.5212 -0.211007763
## Feb 1975  0.5630906 330.5987  0.078254409
## Mar 1975  1.2747939 330.6761 -0.080921048
## Apr 1975  2.3934412 330.7625 -0.015983334
## May 1975  2.8908384 330.8490  0.060204445
## Jun 1975  2.2901417 330.9456  0.184264505
## Jul 1975  0.8516326 331.0422 -0.163862969
## Aug 1975 -1.1543733 331.1452 -0.090845961
## Sep 1975 -2.9275669 331.2482  0.079358661
## Oct 1975 -3.1649278 331.3471 -0.012163030
## Nov 1975 -2.0010387 331.4460 -0.124934662
## Dec 1975 -0.9358567 331.5324 -0.006576687
## Jan 1976 -0.0801751 331.6189  0.041281693
## Feb 1976  0.5630906 331.6925  0.134453115
## Mar 1976  1.2747939 331.7660  0.289186906
## Apr 1976  2.3934412 331.8277  0.188899776
## May 1976  2.8908384 331.8893 -0.070137290
```

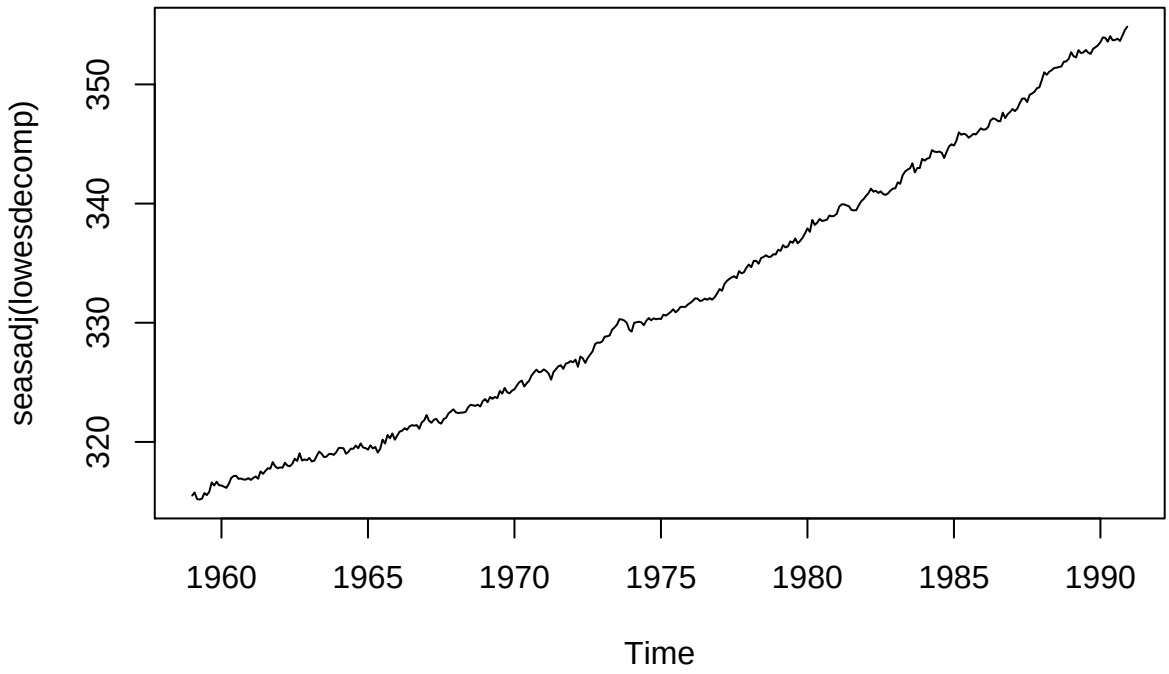
```
## Jun 1976  2.2901417 331.9546 -0.074774405
## Jul 1976  0.8516326 332.0200  0.008400945
## Aug 1976 -1.1543733 332.1048 -0.180380251
## Sep 1976 -2.9275669 332.1895 -0.121973832
## Oct 1976 -3.1649278 332.3133 -0.378400423
## Nov 1976 -2.0010387 332.4371 -0.296076955
## Dec 1976 -0.9358567 332.5974 -0.141553445
## Jan 1977 -0.0801751 332.7577  0.072470471
## Feb 1977  0.5630906 332.9361 -0.249148687
## Mar 1977  1.2747939 333.1144  0.140794524
## Apr 1977  2.3934412 333.2958  0.210799955
## May 1977  2.8908384 333.4771  0.202055449
## Jun 1977  2.2901417 333.6493  0.160586003
## Jul 1977  0.8516326 333.8214  0.086929022
## Aug 1977 -1.1543733 333.9781 -0.233720783
## Sep 1977 -2.9275669 334.1347  0.202817027
## Oct 1977 -3.1649278 334.2749 -0.130015834
## Nov 1977 -2.0010387 334.4151 -0.174098636
## Dec 1977 -0.9358567 334.5497  0.066107256
## Jan 1978 -0.0801751 334.6844  0.195813555
## Feb 1978  0.5630906 334.8168 -0.159871680
## Mar 1978  1.2747939 334.9492  0.246005454
## Apr 1978  2.3934412 335.0675  0.129019188
## May 1978  2.8908384 335.1859 -0.236717014
## Jun 1978  2.2901417 335.2910  0.138840457
## Jul 1978  0.8516326 335.3962  0.122210394
## Aug 1978 -1.1543733 335.4992  0.165199222
## Sep 1978 -2.9275669 335.6022 -0.074624334
## Oct 1978 -3.1649278 335.7083 -0.173372471
## Nov 1978 -2.0010387 335.8144 -0.063370547
## Dec 1978 -0.9358567 335.9257 -0.199866881
## Jan 1979 -0.0801751 336.0370  0.093137192
## Feb 1979  0.5630906 336.1498 -0.122876043
## Mar 1979  1.2747939 336.2625  0.252673091
## Apr 1979  2.3934412 336.3789 -0.062342047
## May 1979  2.8908384 336.4953 -0.086107122
## Jun 1979  2.2901417 336.6219  0.207990780
## Jul 1979  0.8516326 336.7485 -0.040098853
## Aug 1979 -1.1543733 336.8937  0.180626618
## Sep 1979 -2.9275669 337.0390 -0.371460295
## Oct 1979 -3.1649278 337.1995 -0.334557458
## Nov 1979 -2.0010387 337.3599 -0.228904561
## Dec 1979 -0.9358567 337.5252 -0.029342236
## Jan 1980 -0.0801751 337.6905  0.229720495
## Feb 1980  0.5630906 337.8525 -0.225588084
## Mar 1980  1.2747939 338.0145  0.610665707
## Apr 1980  2.3934412 338.1674  0.039186394
## May 1980  2.8908384 338.3202  0.078957146
## Jun 1980  2.2901417 338.4494  0.260433474
## Jul 1980  0.8516326 338.5786 -0.040277732
## Aug 1980 -1.1543733 338.7023 -0.117917970
## Sep 1980 -2.9275669 338.8259 -0.178370593
## Oct 1980 -3.1649278 338.9517  0.053189117
## Nov 1980 -2.0010387 339.0775 -0.146501113
```

```
## Dec 1980 -0.9358567 339.1891 -0.213237304
## Jan 1981 -0.0801751 339.3006 -0.160473089
## Feb 1981  0.5630906 339.3854  0.351477900
## Mar 1981  1.2747939 339.4702  0.464991259
## Apr 1981  2.3934412 339.5470  0.389561584
## May 1981  2.8908384 339.6238  0.225381974
## Jun 1981  2.2901417 339.7096  0.070210321
## Jul 1981  0.8516326 339.7955 -0.327148866
## Aug 1981 -1.1543733 339.8930 -0.468600479
## Sep 1981 -2.9275669 339.9904 -0.542864477
## Oct 1981 -3.1649278 340.1024 -0.257512595
## Nov 1981 -2.0010387 340.2144 -0.023410652
## Dec 1981 -0.9358567 340.3422  0.033661657
## Jan 1982 -0.0801751 340.4699  0.180234373
## Feb 1982  0.5630906 340.5867  0.290194673
## Mar 1982  1.2747939 340.7035  0.551717344
## Apr 1982  2.3934412 340.7854  0.211127238
## May 1982  2.8908384 340.8674  0.201787196
## Jun 1982  2.2901417 340.9207 -0.030807092
## Jul 1982  0.8516326 340.9740  0.054411085
## Aug 1982 -1.1543733 341.0294 -0.225012450
## Sep 1982 -2.9275669 341.0848 -0.357248370
## Oct 1982 -3.1649278 341.1827 -0.327723522
## Nov 1982 -2.0010387 341.2805 -0.189448613
## Dec 1982 -0.9358567 341.4393 -0.183453781
## Jan 1983 -0.0801751 341.5981 -0.317958542
## Feb 1983  0.5630906 341.7891 -0.002143896
## Mar 1983  1.2747939 341.9800 -0.324766880
## Apr 1983  2.3934412 342.1688  0.207747169
## May 1983  2.8908384 342.3577  0.331511281
## Jun 1983  2.2901417 342.5387  0.311121179
## Jul 1983  0.8516326 342.7198  0.238543542
## Aug 1983 -1.1543733 342.8893  0.485062055
## Sep 1983 -2.9275669 343.0588 -0.441231816
## Oct 1983 -3.1649278 343.2109 -0.225959992
## Nov 1983 -2.0010387 343.3630 -0.381938108
## Dec 1983 -0.9358567 343.4967  0.259110725
## Jan 1984 -0.0801751 343.6305 -0.030340036
## Feb 1984  0.5630906 343.7482  0.018683242
## Mar 1984  1.2747939 343.8659 -0.030731109
## Apr 1984  2.3934412 343.9766  0.509944675
## May 1984  2.8908384 344.0873  0.271870522
## Jun 1984  2.2901417 344.1929  0.126941033
## Jul 1984  0.8516326 344.2985  0.069824010
## Aug 1984 -1.1543733 344.4150 -0.150674025
## Sep 1984 -2.9275669 344.5316 -0.703984446
## Oct 1984 -3.1649278 344.6622 -0.327236300
## Nov 1984 -2.0010387 344.7928  0.008261907
## Dec 1984 -0.9358567 344.9304  0.045455306
## Jan 1985 -0.0801751 345.0680 -0.197850888
## Feb 1985  0.5630906 345.2043  0.052606029
## Mar 1985  1.2747939 345.3406  0.634625315
## Apr 1985  2.3934412 345.4620  0.314552011
## May 1985  2.8908384 345.5834  0.275728770
```

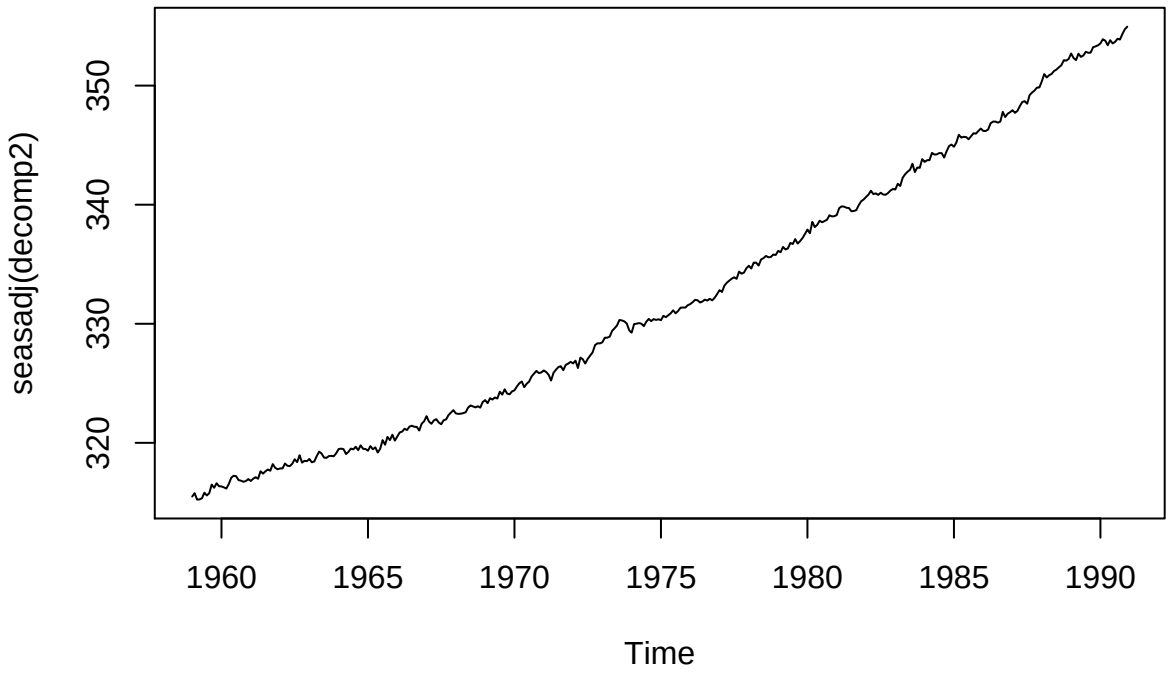
```
## Jun 1985  2.2901417 345.6784  0.101441098
## Jul 1985  0.8516326 345.7734 -0.245034110
## Aug 1985 -1.1543733 345.8496 -0.175217246
## Sep 1985 -2.9275669 345.9258 -0.078212767
## Oct 1985 -3.1649278 346.0122 -0.217233461
## Nov 1985 -2.0010387 346.0985 -0.037504094
## Dec 1985 -0.9358567 346.2120  0.103813561
## Jan 1986 -0.0801751 346.3255 -0.125368378
## Feb 1986  0.5630906 346.4508 -0.223930766
## Mar 1986  1.2747939 346.5761 -0.160930786
## Apr 1986  2.3934412 346.6997  0.286882127
## May 1986  2.8908384 346.8232  0.325945104
## Jun 1986  2.2901417 346.9484  0.141475582
## Jul 1986  0.8516326 347.0735 -0.145181475
## Aug 1986 -1.1543733 347.1967 -0.292294033
## Sep 1986 -2.9275669 347.3198  0.307781023
## Oct 1986 -3.1649278 347.4483 -0.273400331
## Nov 1986 -2.0010387 347.5769 -0.075831625
## Dec 1986 -0.9358567 347.7200 -0.034121999
## Jan 1987 -0.0801751 347.8631  0.077088032
## Feb 1987  0.5630906 348.0232 -0.266321311
## Mar 1987  1.2747939 348.1834 -0.198168285
## Apr 1987  2.3934412 348.3533  0.093211282
## May 1987  2.8908384 348.5233  0.285840913
## Jun 1987  2.2901417 348.7159  0.103918860
## Jul 1987  0.8516326 348.9086 -0.390190728
## Aug 1987 -1.1543733 349.1274 -0.002991728
## Sep 1987 -2.9275669 349.3462 -0.108605113
## Oct 1987 -3.1649278 349.5744 -0.189461752
## Nov 1987 -2.0010387 349.8026 -0.121568332
## Dec 1987 -0.9358567 350.0275 -0.271683634
## Jan 1988 -0.0801751 350.2525  0.117701470
## Feb 1988  0.5630906 350.4673  0.549574858
## Mar 1988  1.2747939 350.6822  0.123010616
## Apr 1988  2.3934412 350.8766  0.179963943
## May 1988  2.8908384 351.0710  0.118167334
## Jun 1988  2.2901417 351.2449  0.124957280
## Jul 1988  0.8516326 351.4188 -0.020440310
## Aug 1988 -1.1543733 351.5659 -0.111496910
## Sep 1988 -2.9275669 351.7129 -0.205365895
## Oct 1988 -3.1649278 351.8467  0.058194331
## Nov 1988 -2.0010387 351.9805 -0.049495383
## Dec 1988 -0.9358567 352.1044  0.041474746
## Jan 1989 -0.0801751 352.2282  0.471945282
## Feb 1989  0.5630906 352.3337  0.033187586
## Mar 1989  1.2747939 352.4392 -0.174007739
## Apr 1989  2.3934412 352.5262  0.350406485
## May 1989  2.8908384 352.6131  0.016070773
## Jun 1989  2.2901417 352.6989 -0.019058509
## Jul 1989  0.8516326 352.7847  0.103624675
## Aug 1989 -1.1543733 352.8858 -0.221445785
## Sep 1989 -2.9275669 352.9869 -0.429328629
## Oct 1989 -3.1649278 353.0951 -0.110128754
## Nov 1989 -2.0010387 353.2032 -0.082178818
```

```
## Dec 1989 -0.9358567 353.3055 -0.019657452
## Jan 1990 -0.0801751 353.4078  0.142364320
## Feb 1990  0.5630906 353.5011  0.445771098
## Mar 1990  1.2747939 353.5945  0.310740245
## Apr 1990  2.3934412 353.6806 -0.094010033
## May 1990  2.8908384 353.7667  0.282489752
## Jun 1990  2.2901417 353.8624 -0.162542318
## Jul 1990  0.8516326 353.9581 -0.229761924
## Aug 1990 -1.1543733 354.0498 -0.215416653
## Sep 1990 -2.9275669 354.1415 -0.493883767
## Oct 1990 -3.1649278 354.2310 -0.146049384
## Nov 1990 -2.0010387 354.3205  0.230535058
## Dec 1990 -0.9358567 354.4114  0.434411556
```

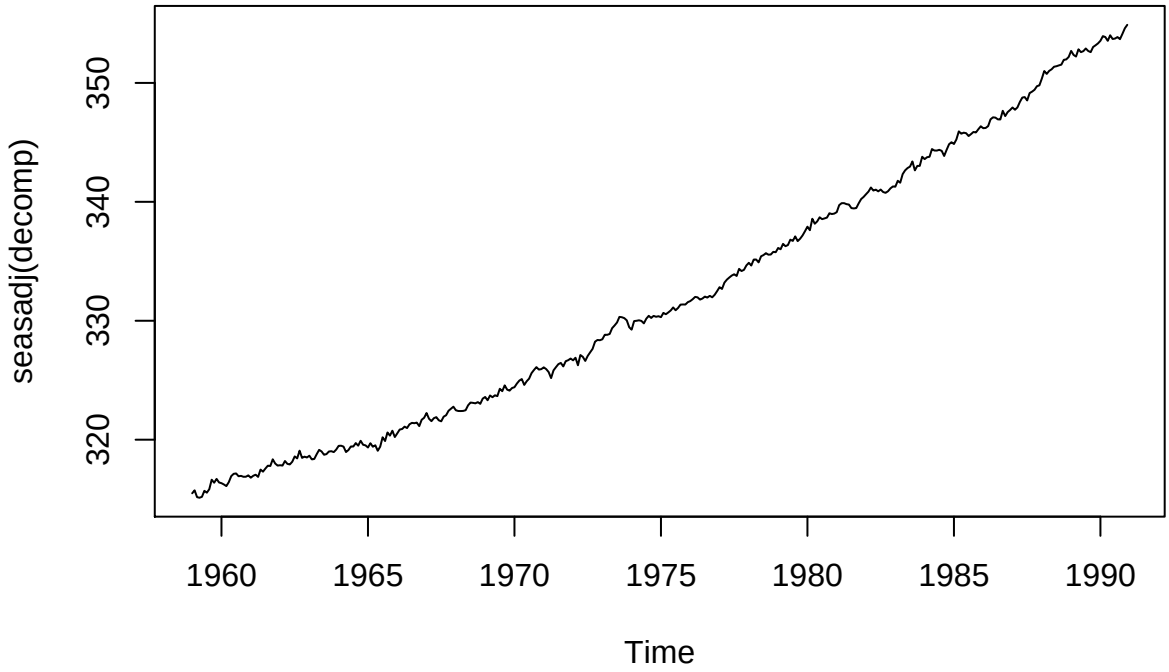
```
plot(seasadj(lowesdecomp))
```



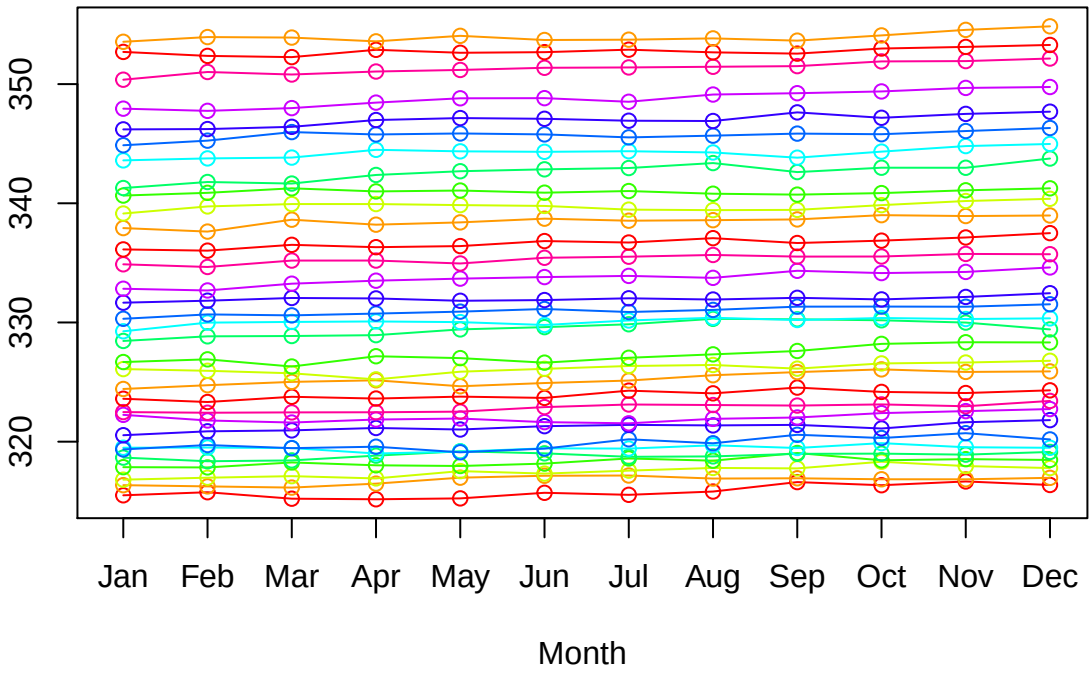

```
plot(seasadj(decomp2))
```



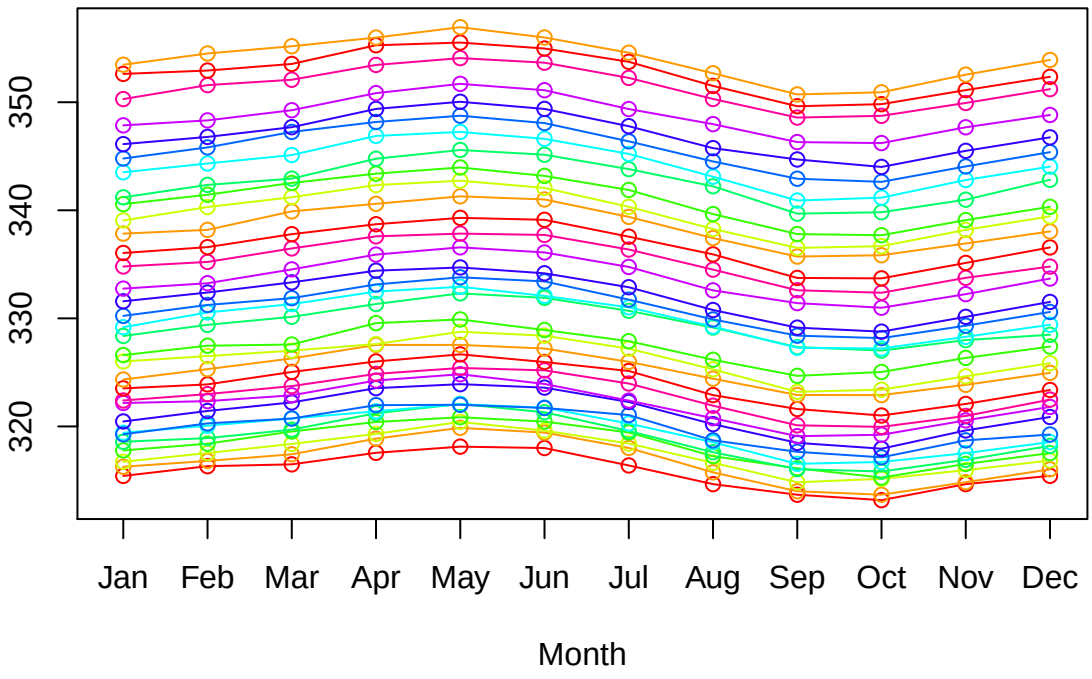

```
plot(seasadj(decomp))
```




```
seasonplot(seasadj(lowesdecomp), col = rainbow(10), main = "seasonal plot")
```




```
seasonplot(hawaii.ts, col = rainbow(10), main = "seasonal plot for original data")
```




```
ndiffs(hawaii.ts) #check how many differencing is needed
```

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

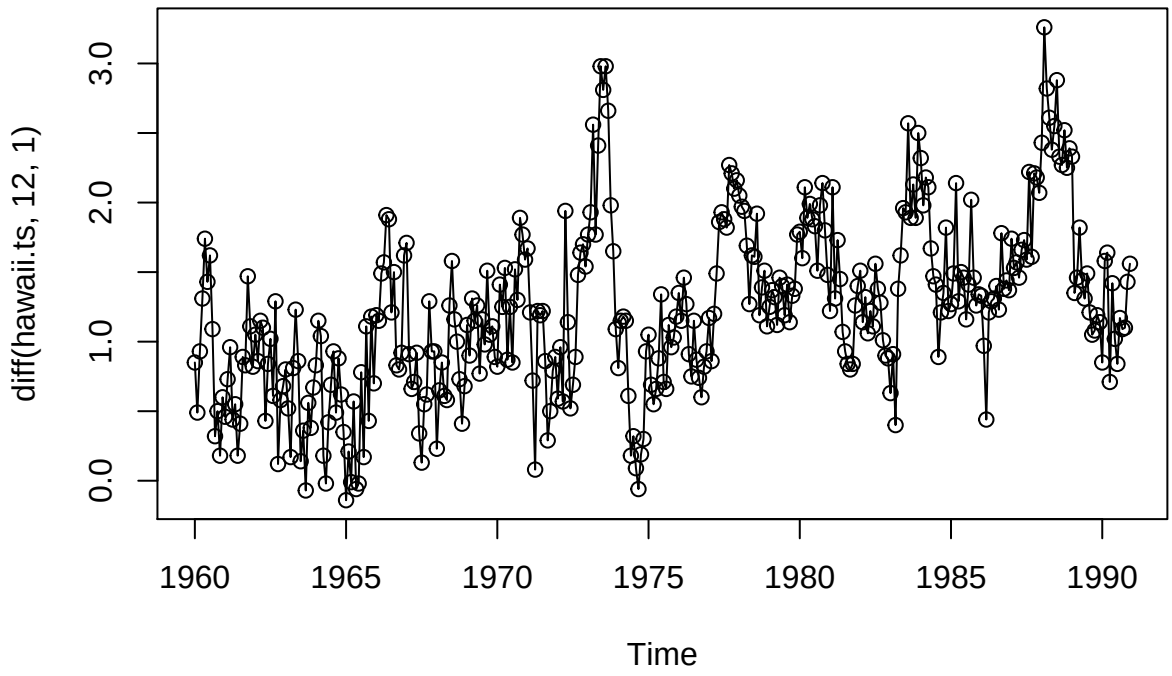
```
ndiffs(seasadj(decomp2))
```

```
## [1] 2
```

```
ndiffs(seasadj(lowesdecomp))
```

```
## [1] 2
```

```
plot(diff(hawaii.ts, 12, 1), type = "o")
```




```
mean(diff(hawaii.ts, 12, 1))
```

```
## [1] 1.229866
```

```
adf.test(diff(hawaii.ts, 12, 1))
```

```
## Warning in adf.test(diff(hawaii.ts, 12, 1)): p-value smaller than printed
## p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(hawaii.ts, 12, 1)
## Dickey-Fuller = -4.9033, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf <- 0 #Initializing vector for adf data
for(i in 1:11){
  adf[i] <- adf.test(diff(diff(hawaii.ts, 12, i)))$statistic
}
```

```
## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value
```

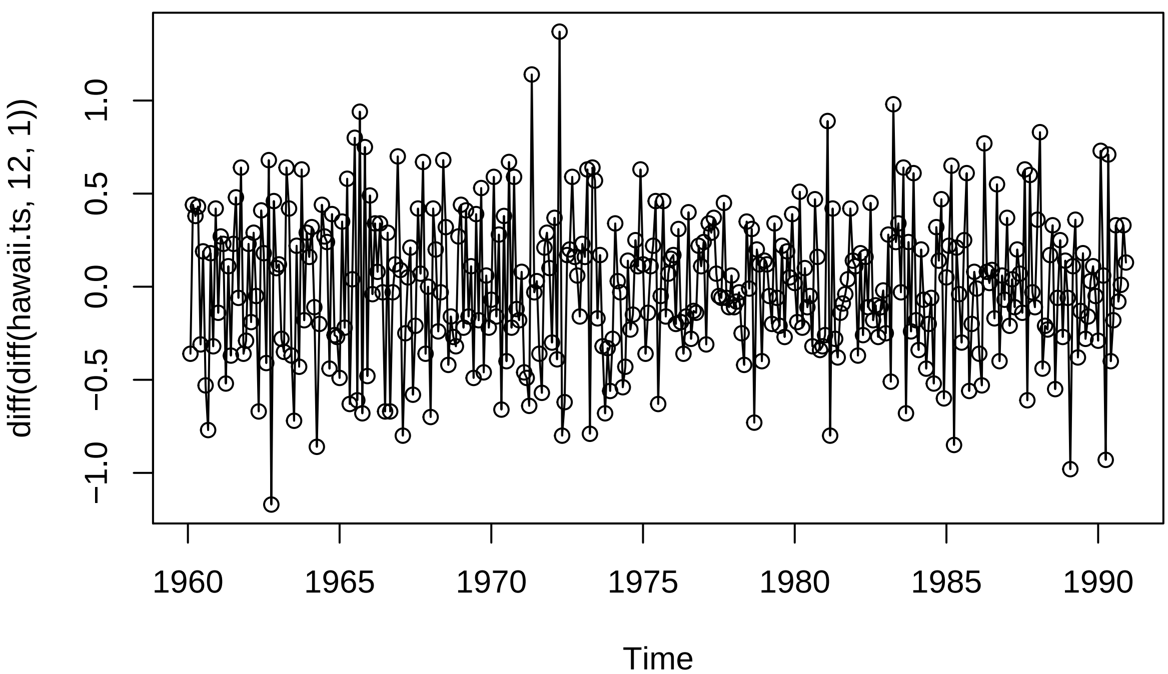
```
## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value
```

```
## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value

## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value

## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value

## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value

## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value

## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value

## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value

## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value

## Warning in adf.test(diff(diff(hawaii.ts, 12, i))): p-value smaller than
## printed p-value
```

```r
which.min(adf) #check which model is the most significant/most stationary
```

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

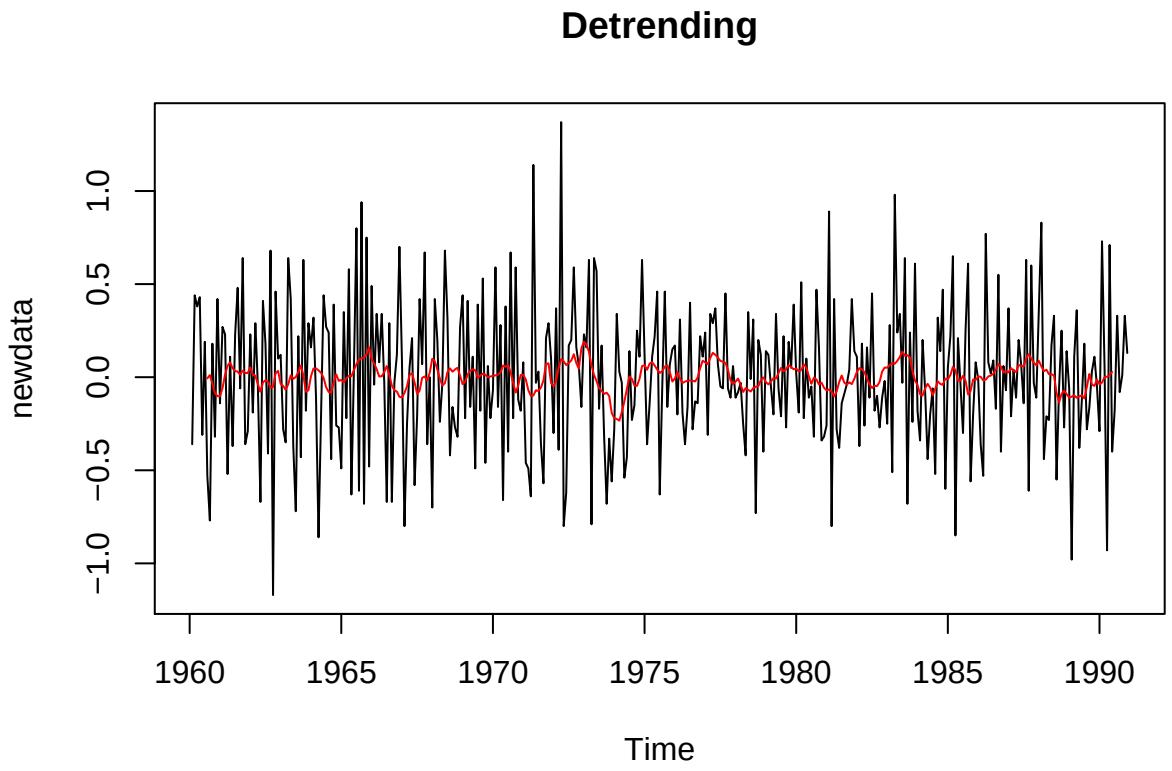
```r
plot(diff(diff(hawaii.ts, 12, 1)), type = "o")
```



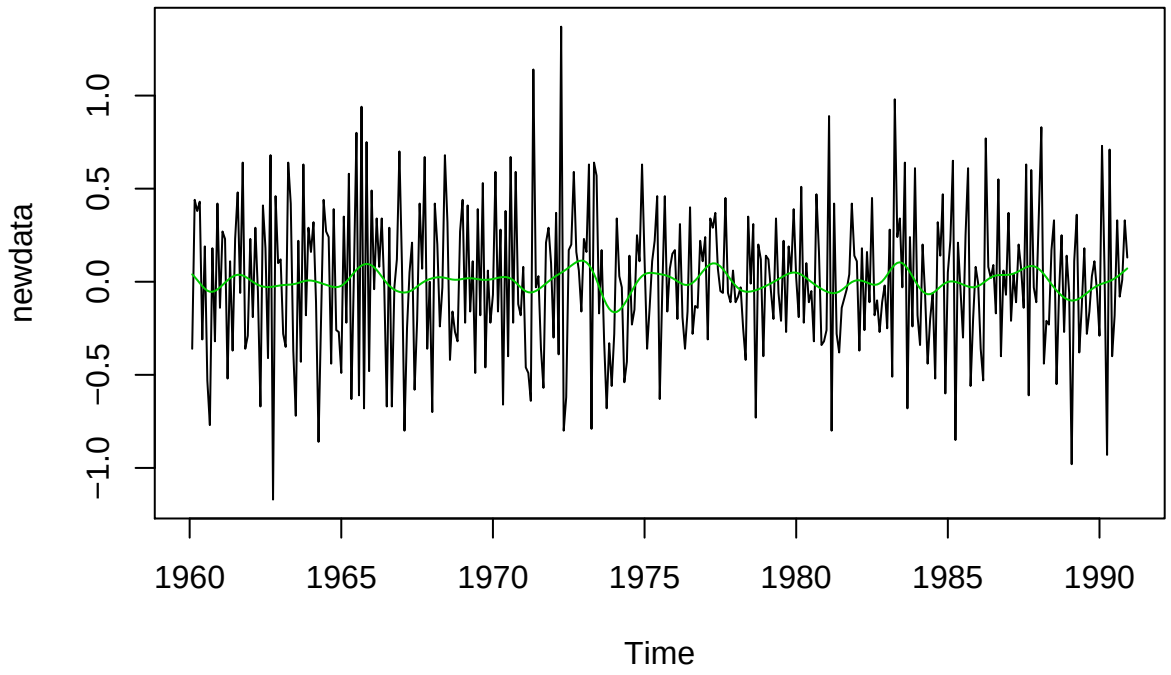
```r
mean(diff(diff(hawaii.ts, 12, 1)))
```

```
## [1] 0.001913747
```

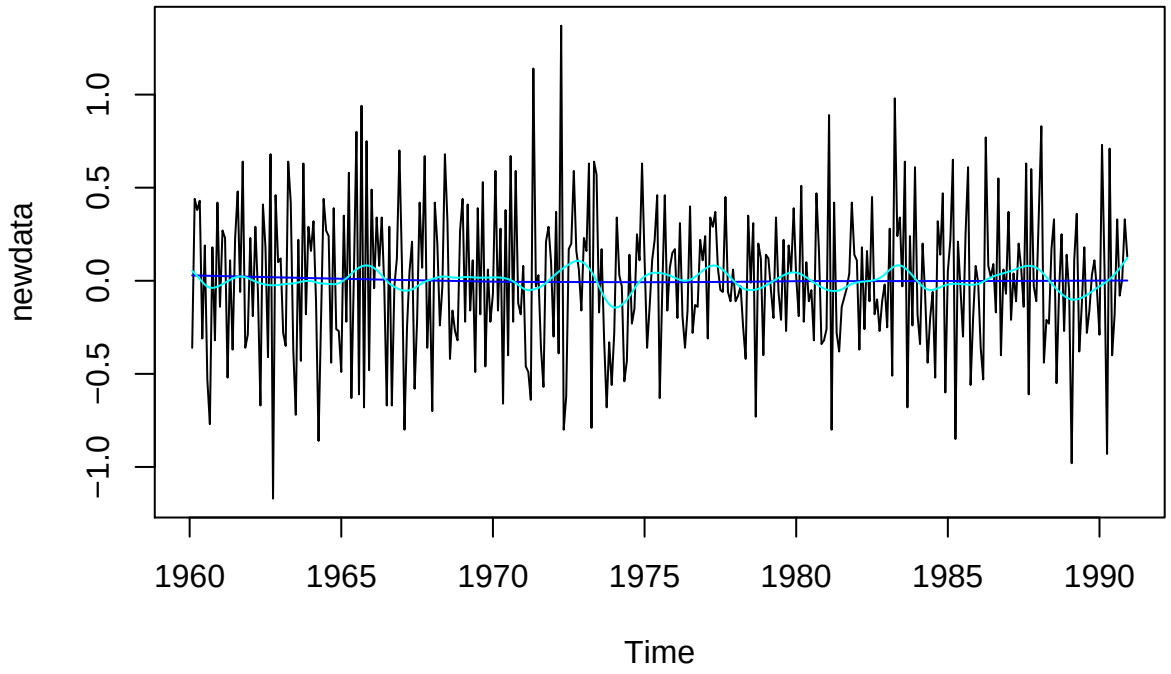
```r
#It is a new dataset after 1st and seasonal differencings
newdata <- diff(diff(hawaii.ts, 12, 1))


#Moving average smoothings/filtering
mv <- stats::filter(newdata, sides = 2, filter = c(0.5, rep(1, 11), 0.5)/12)
plot(newdata, main = "Detrending")
lines(mv, col = 2)
```



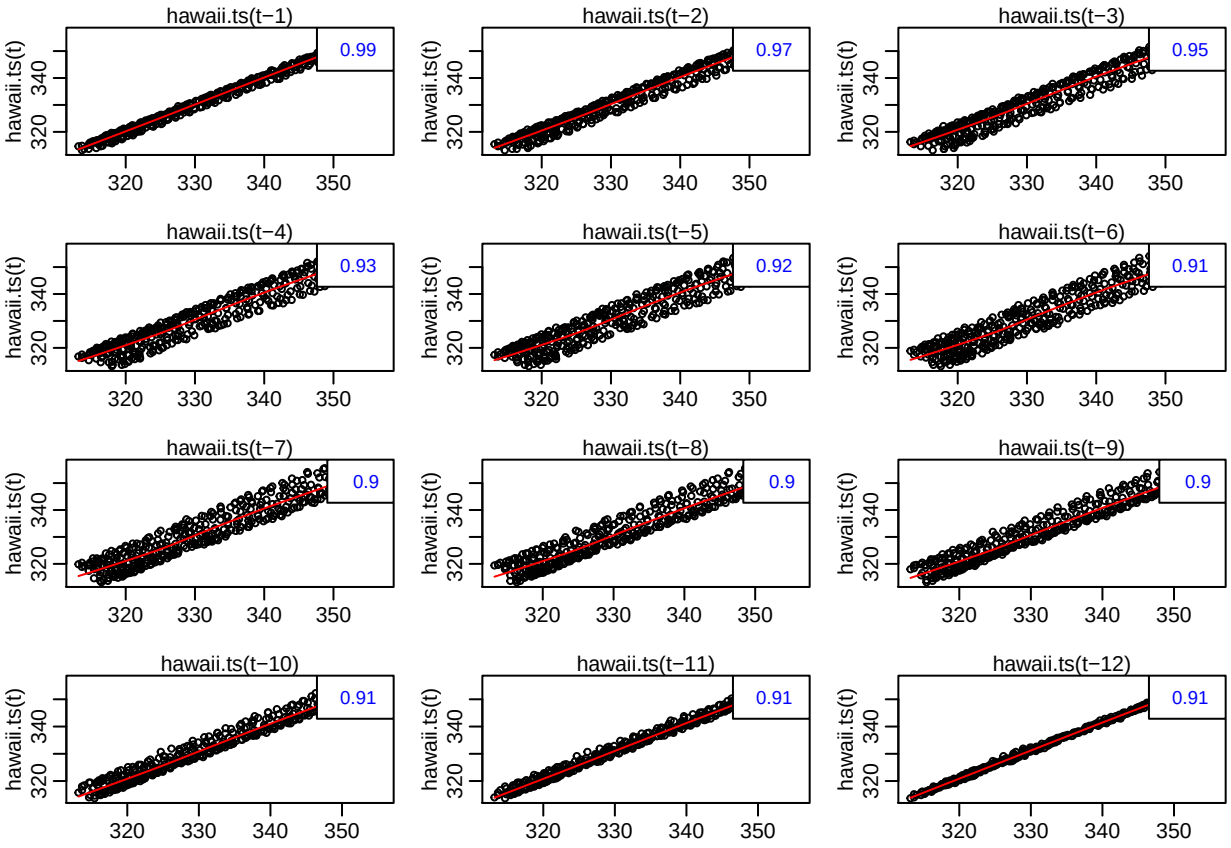

```r
#Kernel smooth
plot(newdata)
lines(ksmooth(time(newdata), newdata, kernel = "normal", bandwidth = 1), col = 3)
```



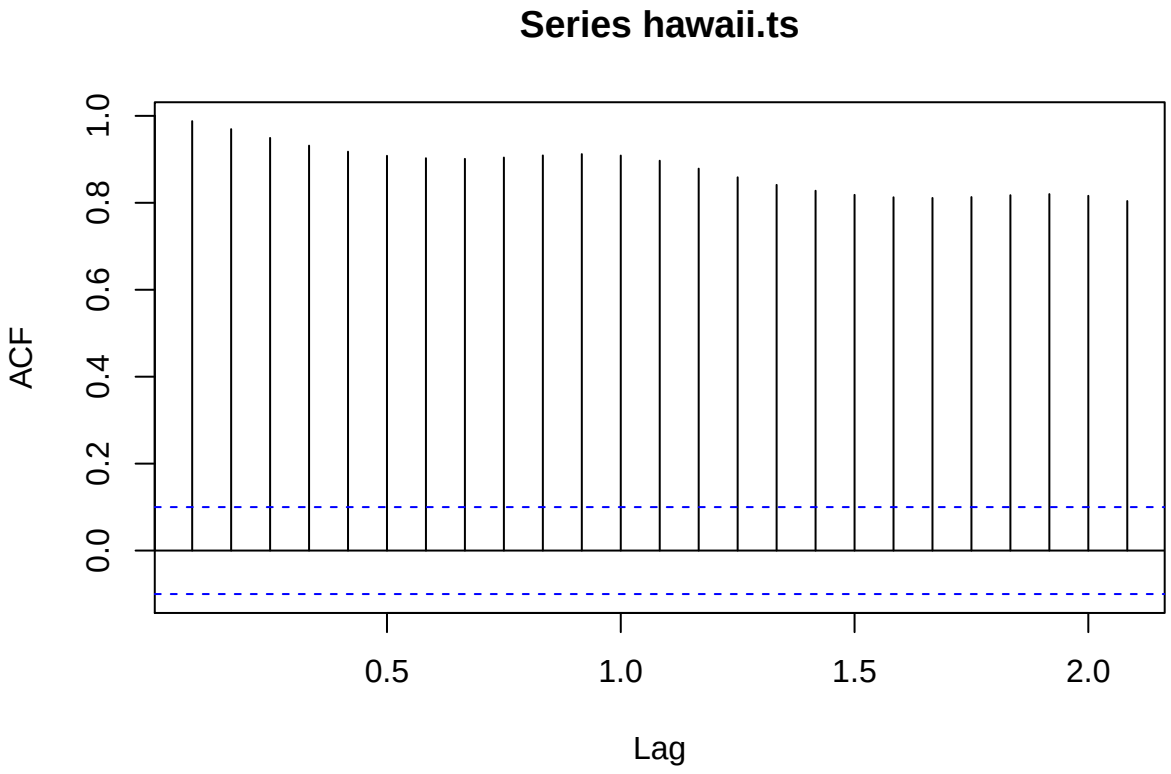

```
#Lowess
plot(newdata)
lines(lowess(newdata), col = 4)

#smooth splines
lines(smooth.spline(time(newdata), newdata, spar = 0.5), col = 5)
```



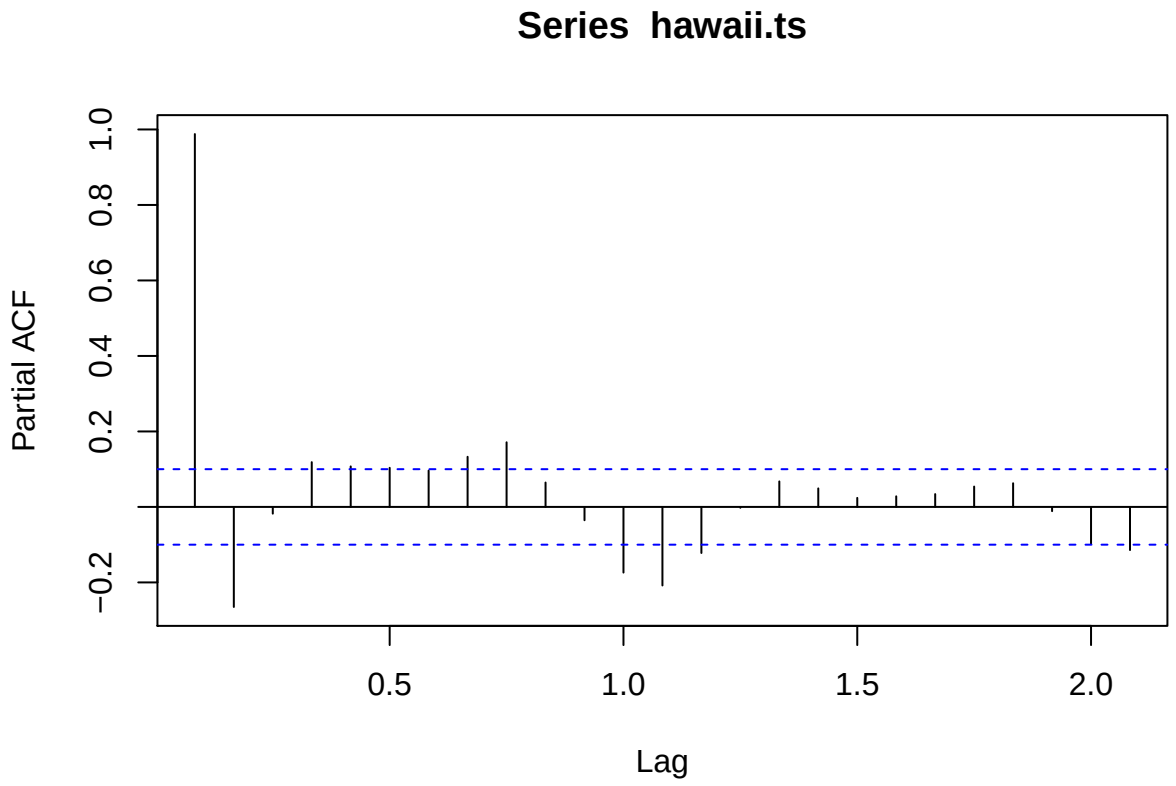

```
#Scatterplot matrix and acf and pacf for sanity check
lag1.plot(hawaii.ts, max.lag = 12)
```



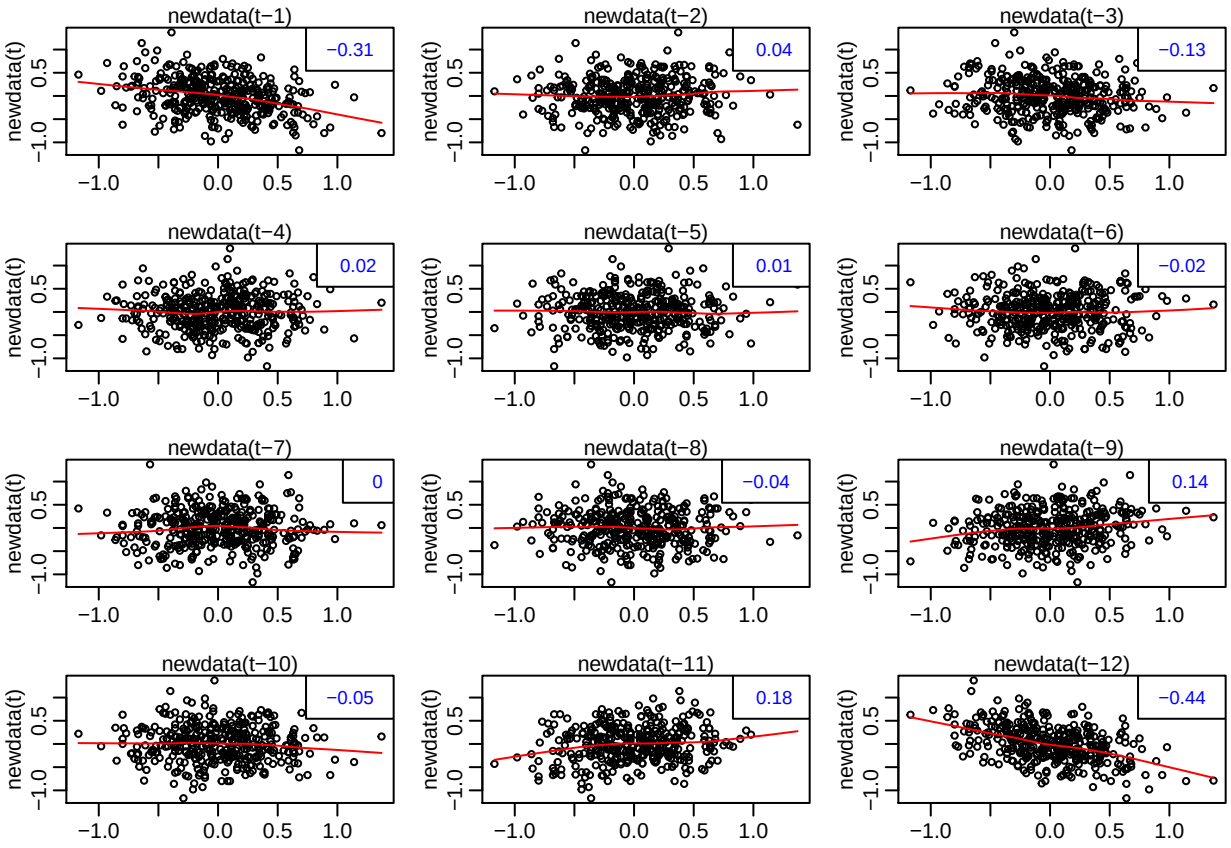

```
acf(hawaii.ts)
```



```
pacf(hawaii.ts)
```




```
lag1.plot(newdata, max.lag = 12)
```



Comment:

We plotted our original data first, and in general, and we could easily find out that there is a very consistent and obvious seasonal pattern with linear trend. When we investigate the data, there is a obvious pattern each year that CO2 is increasing at the beginnin of the year upto May, decreasing from May to September/October, and later increasing again at end of the year. So, we end up saying that there will be an overall increasing trend and seasonal effect. Also, it became more obvious when we performed Dickey-Fuller Test, as the p-value of the test was 0.3964.

To remove the seasonal trend, we take seasonal difference of differencing lag of 12, and then, it looked pretty stationary (and Dickey-Fuller test also back this up); however, we found that as we differenced more, we could make the model more stationary. So, we ran the for-loop for differencing lag of 1 to 11, and found that when diffencing an additional 1st difference operation makes the model the most stationary. (but, remember that since the data is trasnsformed, some of the years were cut-off/removed)

We also tried to use the function "decompose" (both additive and multiplicative models) and "stl" to see whether we could determine the trend using a moving average and lowess. Here, since the seaosonal variation is relatively constant, it is better for us to use additive model. From these two, we could easily found the trend is increasing.

Last but not least, we tried to plot a scatterplot matrix to back-up what we did above. As it can be see, every sample autocorrelation is really high and we found really strong positive linear relationships at every lag. And, after we transformed the data (chased the staionary), we could finally conclude that our transformation was the right decision, as we have seen less autocorrelation and lowess fit looks much less linear.

(S)Arima

```
#Split the train and testing
test <- tail(hawaii.ts, 10)
train <- head(hawaii.ts, -10)

length(test)
```

```
## [1] 10
```

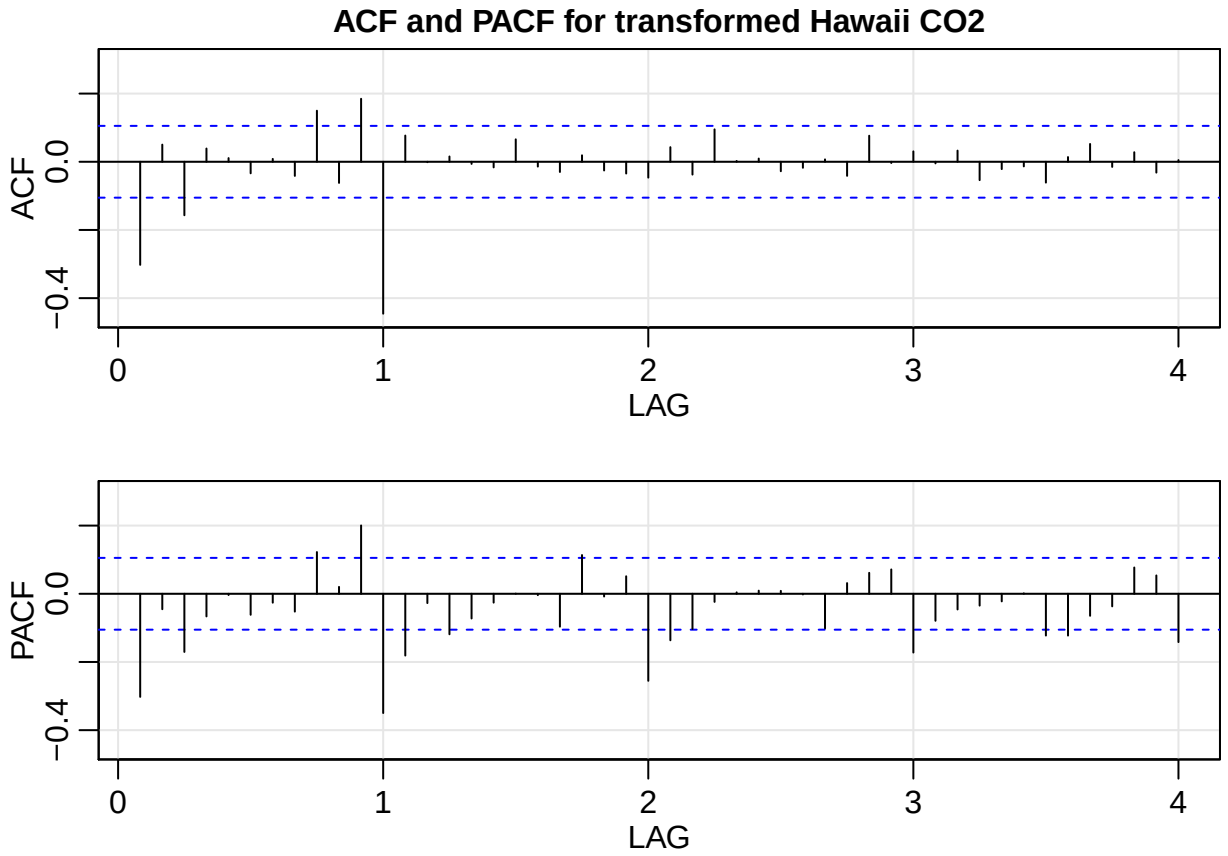
```
length(train)
```

```
## [1] 374
```

```
length(hawaii.ts)
```

```
## [1] 384
```

```
newtrain <- diff(diff(train, 12, 1))




#ACF, PACF, EACF
acf2(newtrain, main = "ACF and PACF for transformed Hawaii CO2")
```




```
##          ACF  PACF
##  [1,] -0.30 -0.30
```

```
##  [2,]  0.05 -0.05
##  [3,] -0.16 -0.17
##  [4,]  0.04 -0.07
##  [5,]  0.01  0.00
##  [6,] -0.03 -0.06
##  [7,]  0.01 -0.03
##  [8,] -0.04 -0.05
##  [9,]  0.15  0.12
## [10,] -0.06  0.02
## [11,]  0.18  0.20
## [12,] -0.45 -0.35
## [13,]  0.08 -0.18
## [14,]  0.00 -0.03
## [15,]  0.02 -0.12
## [16,] -0.01 -0.07
## [17,] -0.02 -0.03
## [18,]  0.07  0.00
## [19,] -0.01  0.00
## [20,] -0.03 -0.10
## [21,]  0.02  0.11
## [22,] -0.03 -0.01
## [23,] -0.03  0.05
## [24,] -0.05 -0.26
## [25,]  0.04 -0.14
## [26,] -0.04 -0.10
## [27,]  0.10 -0.02
## [28,]  0.00  0.00
## [29,]  0.01  0.01
## [30,] -0.03  0.01
## [31,] -0.02  0.00
## [32,]  0.01 -0.10
## [33,] -0.04  0.03
## [34,]  0.08  0.06
## [35,]  0.00  0.07
## [36,]  0.03 -0.17
## [37,]  0.00 -0.08
## [38,]  0.03 -0.05
## [39,] -0.05 -0.03
## [40,] -0.02 -0.02
## [41,] -0.01  0.00
## [42,] -0.06 -0.12
## [43,]  0.01 -0.12
## [44,]  0.05 -0.06
## [45,] -0.02 -0.04
## [46,]  0.03  0.08
## [47,] -0.03  0.05
## [48,]  0.00 -0.14
```

```
eacf(newtrain)
```

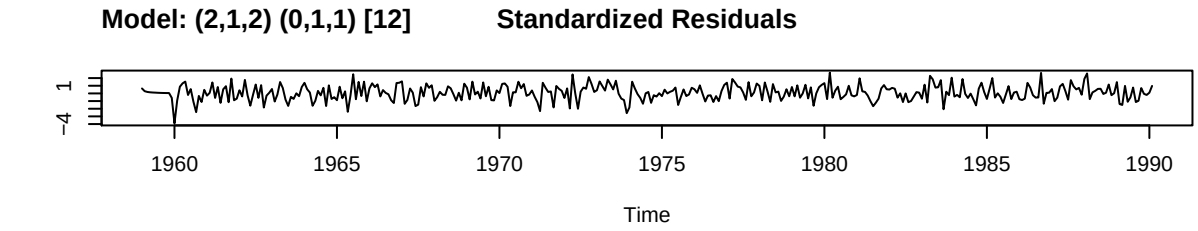
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o x o o o o o x o x  x  o  o
## 1 x o x o o o o o x o o  x  o  o
## 2 x x x o o o o o o o o  x  x  o
```

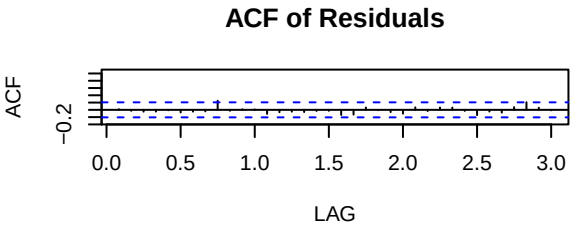
```
## 3 x x x o o o o o o o o  x  x  x
## 4 o x x o o o o o o o o  x  o  o
## 5 o x o x o o o o o o o  x  x  o
## 6 x x o x o o o o o o o  x  o  x
## 7 x x x o o o o o o o o  x  o  o
```

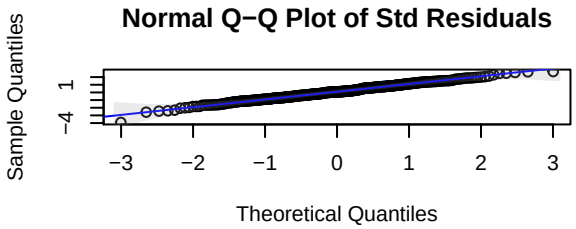
```
#Sarima - model comparison (out = not appropriate, good = appropriate)

sarima(train, 2, 1, 2, 0, 1, 1, 12) #out
```

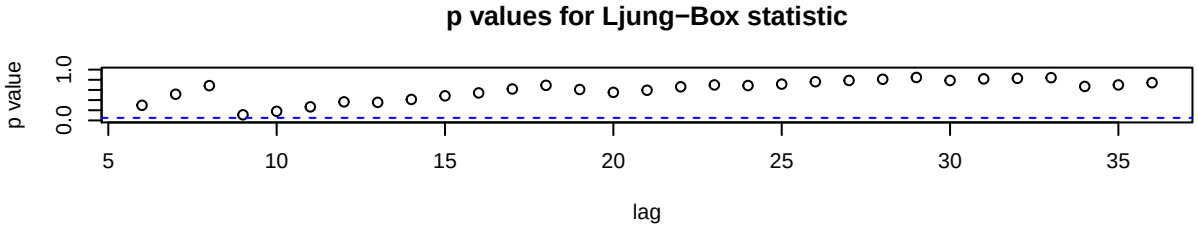
```
## initial  value -0.935835
## iter   2 value -1.110181
## iter   3 value -1.151552
## iter   4 value -1.195525
## iter   5 value -1.209348
## iter   6 value -1.223039
## iter   7 value -1.225188
## iter   8 value -1.226095
## iter   9 value -1.229225
## iter  10 value -1.230687
## iter  11 value -1.231082
## iter  12 value -1.231084
## iter  13 value -1.231097
## iter  14 value -1.231109
## iter  15 value -1.231129
## iter  16 value -1.231165
## iter  17 value -1.231214
## iter  18 value -1.231259
## iter  19 value -1.231283
## iter  20 value -1.231306
## iter  21 value -1.231328
## iter  22 value -1.231337
## iter  23 value -1.231338
## iter  23 value -1.231338
## final  value -1.231338
## converged
## initial  value -1.241082
## iter   2 value -1.243571
## iter   3 value -1.244730
## iter   4 value -1.245268
## iter   5 value -1.245320
## iter   6 value -1.245362
## iter   7 value -1.245363
## iter   8 value -1.245365
## iter   9 value -1.245374
## iter  10 value -1.245391
## iter  11 value -1.245423
## iter  12 value -1.245454
## iter  13 value -1.245472
## iter  14 value -1.245475
## iter  15 value -1.245477
## iter  16 value -1.245478
## iter  17 value -1.245479
## iter  18 value -1.245480
## iter  19 value -1.245480
```

```
## iter  20 value -1.245480
## iter  21 value -1.245480
## iter  22 value -1.245480
## iter  23 value -1.245481
## iter  24 value -1.245481
## iter  25 value -1.245482
## iter  26 value -1.245482
## iter  27 value -1.245482
## iter  28 value -1.245482
## iter  29 value -1.245482
## iter  30 value -1.245482
## iter  31 value -1.245482
## iter  32 value -1.245483
## iter  33 value -1.245483
## iter  34 value -1.245484
## iter  35 value -1.245484
## iter  36 value -1.245484
## iter  36 value -1.245484
## iter  36 value -1.245484
## final  value -1.245484
## converged
```









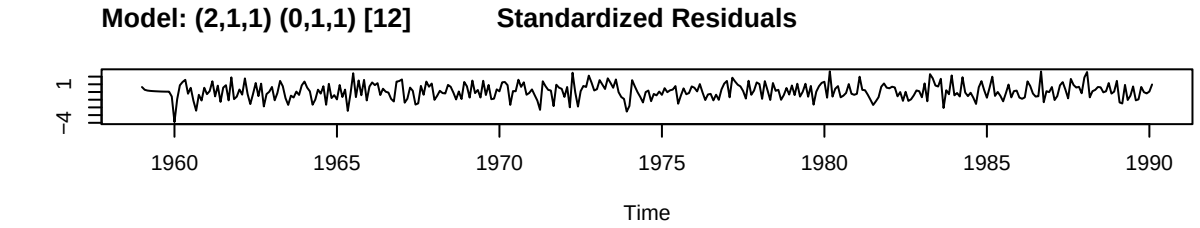

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
```

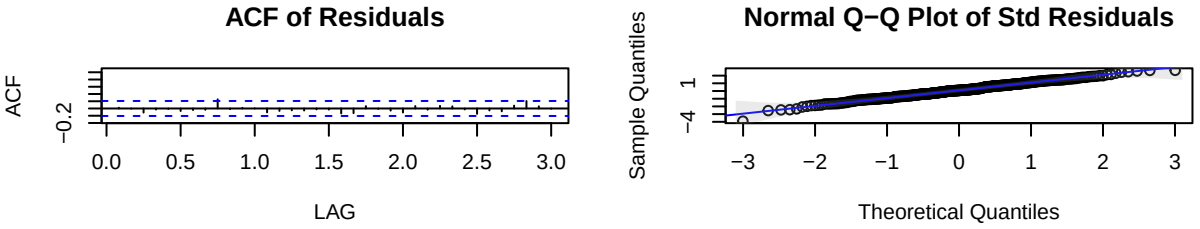
```
## Coefficients:
##           ar1     ar2      ma1      ma2     sma1
##       -0.0428  0.2566  -0.3112  -0.2909  -0.8548
## s.e.   0.4182  0.1354   0.4197   0.2463   0.0324
##
## sigma^2 estimated as 0.07925:  log likelihood = -62.62,  aic = 137.23
##
## $degrees_of_freedom
## [1] 356
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.0428 0.4182  -0.1024  0.9185
## ar2    0.2566 0.1354   1.8957  0.0588
## ma1   -0.3112 0.4197  -0.7414  0.4589
## ma2   -0.2909 0.2463  -1.1809  0.2384
## sma1  -0.8548 0.0324 -26.3509  0.0000
##
## $AIC
## [1] -1.508452
##
## $AICc
## [1] -1.502492
##
## $BIC
## [1] -2.455988
```

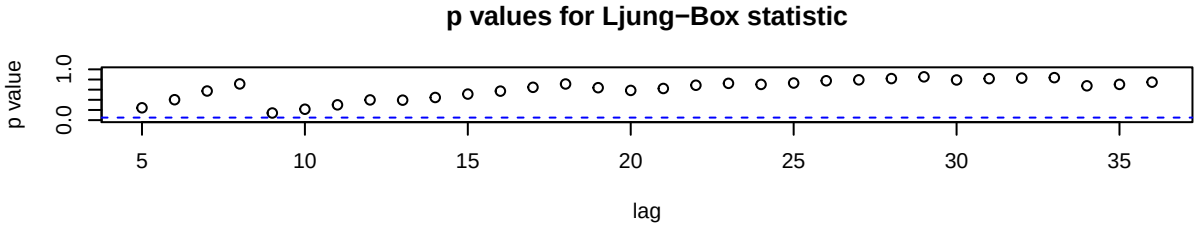
```
sarima(train, 2, 1, 1, 0, 1, 1, 12) #good
```

```
## initial  value -0.935835
## iter   2 value -1.119234
## iter   3 value -1.154297
## iter   4 value -1.193102
## iter   5 value -1.207932
## iter   6 value -1.219985
## iter   7 value -1.220823
## iter   8 value -1.222039
## iter   9 value -1.222535
## iter  10 value -1.222907
## iter  11 value -1.223106
## iter  12 value -1.223593
## iter  13 value -1.225369
## iter  14 value -1.227915
## iter  15 value -1.228375
## iter  16 value -1.229311
## iter  17 value -1.229518
## iter  18 value -1.229827
## iter  19 value -1.229880
## iter  20 value -1.229891
## iter  21 value -1.229900
## iter  22 value -1.229900
## iter  23 value -1.229900
## iter  24 value -1.229900
## iter  24 value -1.229900
## iter  24 value -1.229900
```

```
## final  value -1.229900
## converged
## initial  value -1.239498
## iter   2 value -1.242061
## iter   3 value -1.243259
## iter   4 value -1.243892
## iter   5 value -1.243949
## iter   6 value -1.244042
## iter   7 value -1.244252
## iter   8 value -1.244432
## iter   9 value -1.244479
## iter  10 value -1.244482
## iter  10 value -1.244482
## final  value -1.244482
## converged
```







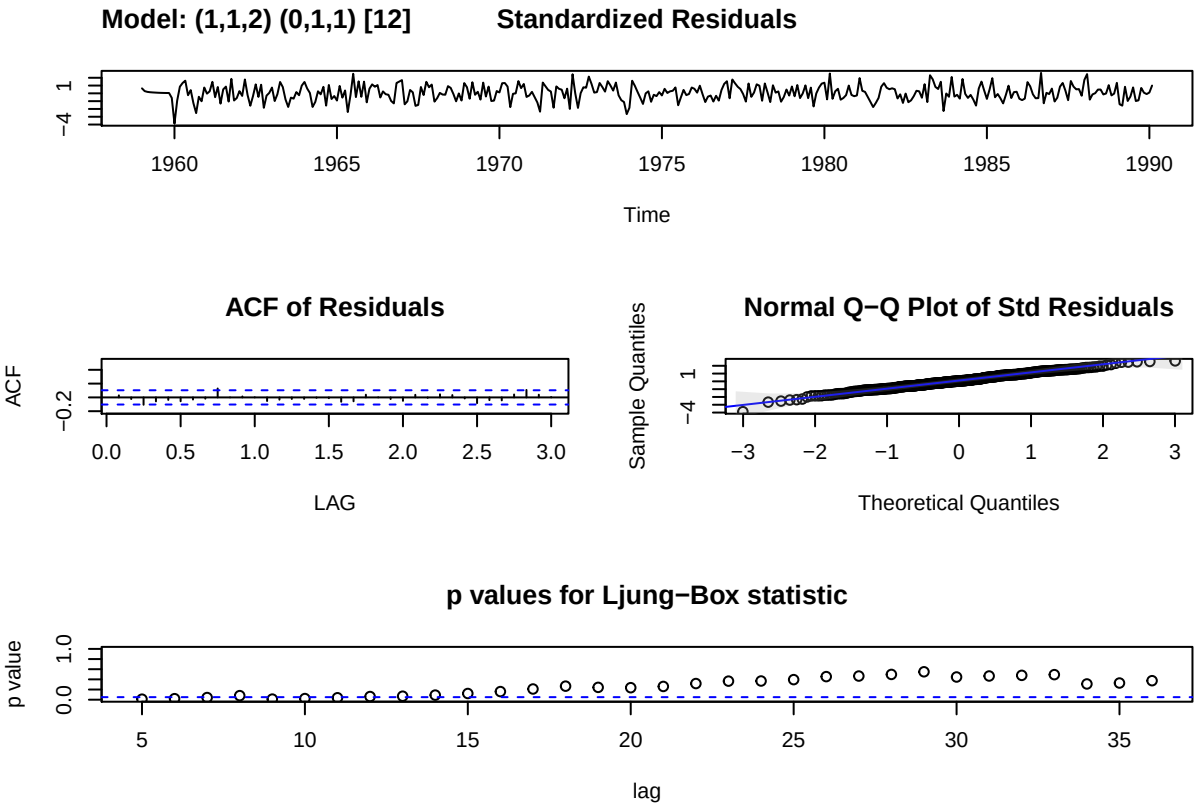

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ar2      ma1     sma1
##       0.3994  0.1131  -0.7566  -0.8544
## s.e.  0.1354  0.0784   0.1222   0.0323
##
## sigma^2 estimated as 0.07941:  log likelihood = -62.98,  aic = 135.96
```

```
##
## $degrees_of_freedom
## [1] 357
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.3994 0.1354   2.9487  0.0034
## ar2    0.1131 0.0784   1.4420  0.1502
## ma1   -0.7566 0.1222  -6.1893  0.0000
## sma1  -0.8544 0.0323 -26.4544  0.0000
##
## $AIC
## [1] -1.511775
##
## $AICc
## [1] -1.505992
##
## $BIC
## [1] -2.469804
```

```
sarima(train, 1, 1, 2, 0, 1, 1, 12) #out
```

```
## initial  value -0.935476
## iter   2 value -1.109793
## iter   3 value -1.146000
## iter   4 value -1.180315
## iter   5 value -1.194590
## iter   6 value -1.208432
## iter   7 value -1.208931
## iter   8 value -1.210368
## iter   9 value -1.210728
## iter  10 value -1.210869
## iter  11 value -1.210973
## iter  12 value -1.211334
## iter  13 value -1.213761
## iter  14 value -1.215596
## iter  15 value -1.216939
## iter  16 value -1.217021
## iter  17 value -1.217202
## iter  18 value -1.217216
## iter  19 value -1.217222
## iter  20 value -1.217222
## iter  20 value -1.217222
## iter  20 value -1.217222
## final  value -1.217222
## converged
## initial  value -1.232349
## iter   2 value -1.235948
## iter   3 value -1.236708
## iter   4 value -1.238614
## iter   5 value -1.238764
## iter   6 value -1.238793
## iter   7 value -1.238799
## iter   7 value -1.238799
## iter   7 value -1.238799
```

```
## final  value -1.238799
## converged
```

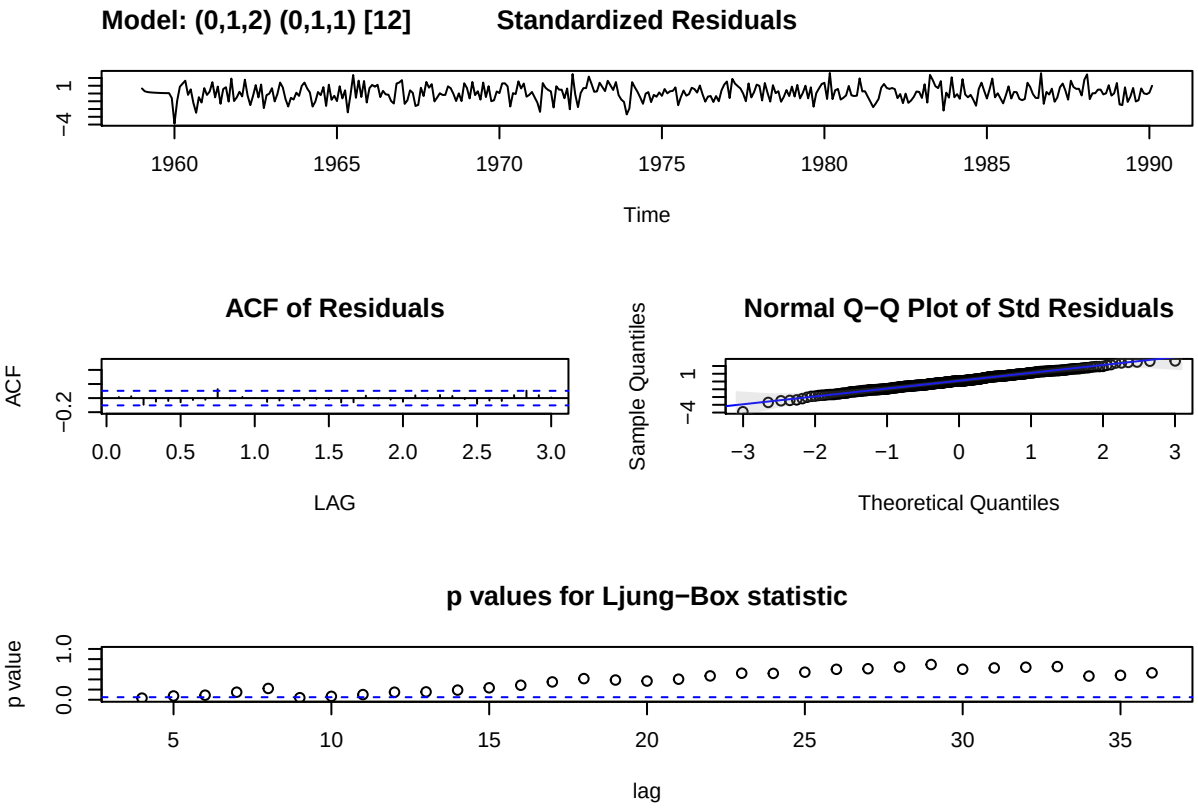



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ma1      ma2     sma1
##       -0.7194  0.3546  -0.2583  -0.8566
## s.e.   0.9808  0.9844   0.3870   0.0324
##
## sigma^2 estimated as 0.0803:  log likelihood = -65.03,  aic = 140.06
##
## $degrees_of_freedom
## [1] 357
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.7194 0.9808  -0.7335  0.4637
## ma1    0.3546 0.9844   0.3602  0.7189
## ma2   -0.2583 0.3870  -0.6673  0.5050
## sma1  -0.8566 0.0324 -26.4166  0.0000
##
## $AIC
## [1] -1.500574
```

```
##
## $AICc
## [1] -1.49479
##
## $BIC
## [1] -2.458603
```

```
sarima(train, 0, 1, 2, 0, 1, 1, 12) #out
```

```
## initial  value -0.935696
## iter   2 value -1.145231
## iter   3 value -1.185777
## iter   4 value -1.206813
## iter   5 value -1.212537
## iter   6 value -1.214244
## iter   7 value -1.215755
## iter   8 value -1.215896
## iter   9 value -1.215902
## iter  10 value -1.215905
## iter  10 value -1.215905
## iter  10 value -1.215905
## final  value -1.215905
## converged
## initial  value -1.233924
## iter   2 value -1.237394
## iter   3 value -1.238986
## iter   4 value -1.240146
## iter   5 value -1.240229
## iter   6 value -1.240231
## iter   6 value -1.240231
## iter   6 value -1.240231
## final  value -1.240231
## converged
```

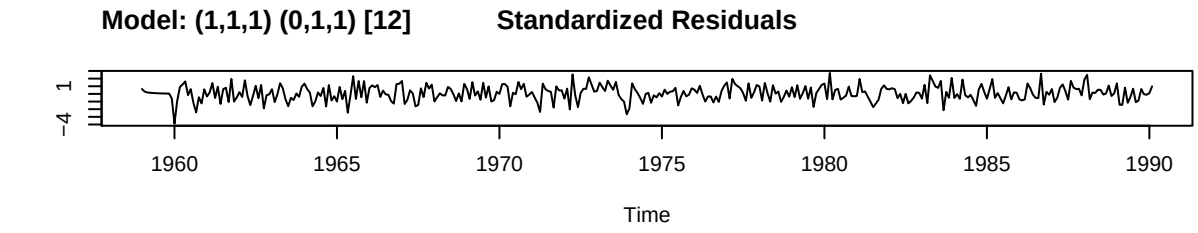


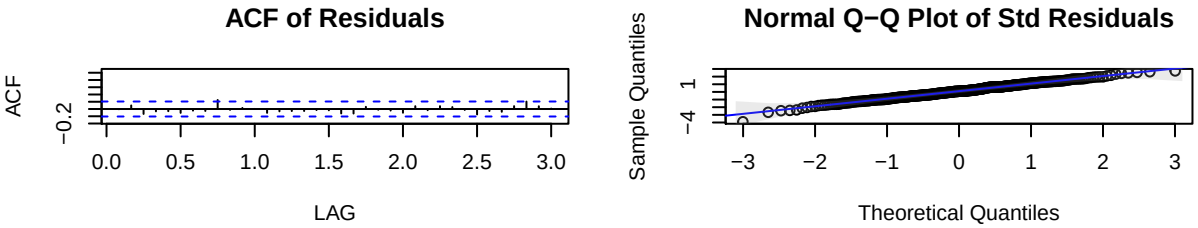

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ma1      ma2     sma1
##       -0.3570  -0.0545  -0.8542
## s.e.   0.0527   0.0531   0.0322
##
## sigma^2 estimated as 0.08011:  log likelihood = -64.51,  aic = 137.03
##
## $degrees_of_freedom
## [1] 358
##
## $ttable
##      Estimate     SE  t.value p.value
## ma1   -0.3570 0.0527  -6.7703  0.0000
## ma2   -0.0545 0.0531  -1.0264  0.3054
## sma1  -0.8542 0.0322 -26.5379  0.0000
##
## $AIC
## [1] -1.508341
##
## $AICc
## [1] -1.502704
```

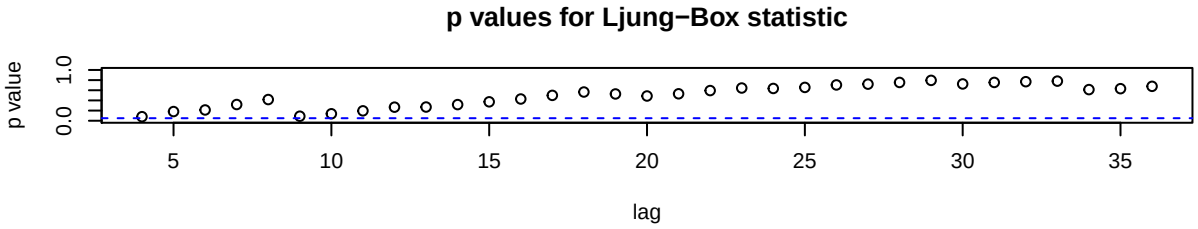
```
##
## $BIC
## [1] -2.476863
```

```
sarima(train, 1, 1, 1, 0, 1, 1, 12) #good
```

```
## initial  value -0.935476
## iter   2 value -1.116989
## iter   3 value -1.149832
## iter   4 value -1.180751
## iter   5 value -1.193873
## iter   6 value -1.205496
## iter   7 value -1.206739
## iter   8 value -1.207515
## iter   9 value -1.207585
## iter  10 value -1.207613
## iter  11 value -1.207620
## iter  12 value -1.207622
## iter  13 value -1.207622
## iter  14 value -1.207624
## iter  15 value -1.207624
## iter  16 value -1.207625
## iter  17 value -1.207625
## iter  17 value -1.207625
## iter  17 value -1.207625
## final  value -1.207625
## converged
## initial  value -1.227745
## iter   2 value -1.228668
## iter   3 value -1.235103
## iter   4 value -1.236139
## iter   5 value -1.236575
## iter   6 value -1.237112
## iter   7 value -1.239418
## iter   8 value -1.240757
## iter   9 value -1.241160
## iter  10 value -1.241257
## iter  11 value -1.241318
## iter  12 value -1.241447
## iter  13 value -1.241582
## iter  14 value -1.241620
## iter  15 value -1.241626
## iter  16 value -1.241626
## iter  16 value -1.241626
## iter  16 value -1.241626
## final  value -1.241626
## converged
```





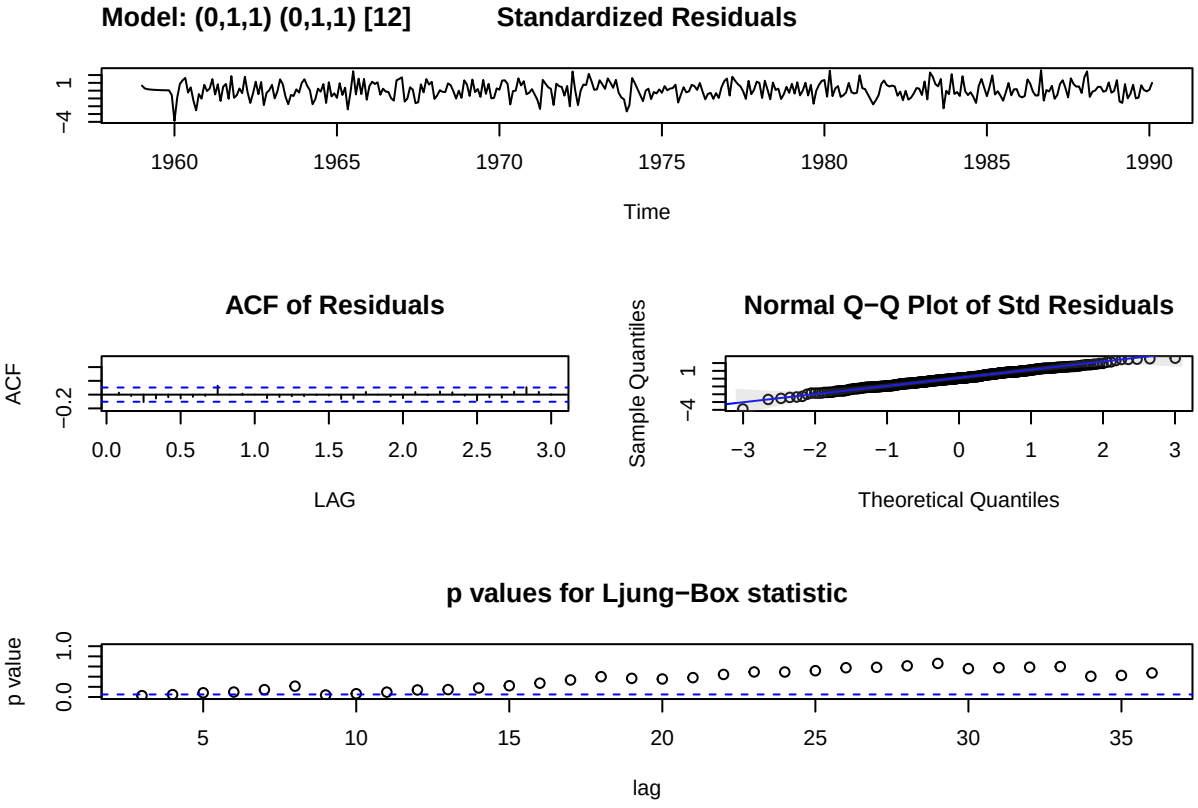



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ma1     sma1
##       0.2522  -0.5953  -0.8540
## s.e.  0.1497   0.1279   0.0321
##
## sigma^2 estimated as 0.07988:  log likelihood = -64.01,  aic = 136.02
##
## $degrees_of_freedom
## [1] 358
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.2522 0.1497   1.6847  0.0929
## ma1   -0.5953 0.1279  -4.6557  0.0000
## sma1  -0.8540 0.0321 -26.6456  0.0000
##
## $AIC
## [1] -1.511151
##
## $AICc
## [1] -1.505513
```

```
##
## $BIC
## [1] -2.479673
```

```
sarima(train, 0, 1, 1, 0, 1, 1, 12) #good
```

```
## initial  value -0.935696
## iter   2 value -1.149414
## iter   3 value -1.183278
## iter   4 value -1.205436
## iter   5 value -1.211805
## iter   6 value -1.212611
## iter   7 value -1.214642
## iter   8 value -1.214790
## iter   9 value -1.214798
## iter  10 value -1.214798
## iter  10 value -1.214798
## iter  10 value -1.214798
## final  value -1.214798
## converged
## initial  value -1.232451
## iter   2 value -1.235908
## iter   3 value -1.237512
## iter   4 value -1.238703
## iter   5 value -1.238787
## iter   6 value -1.238789
## iter   6 value -1.238789
## final  value -1.238789
## converged
```

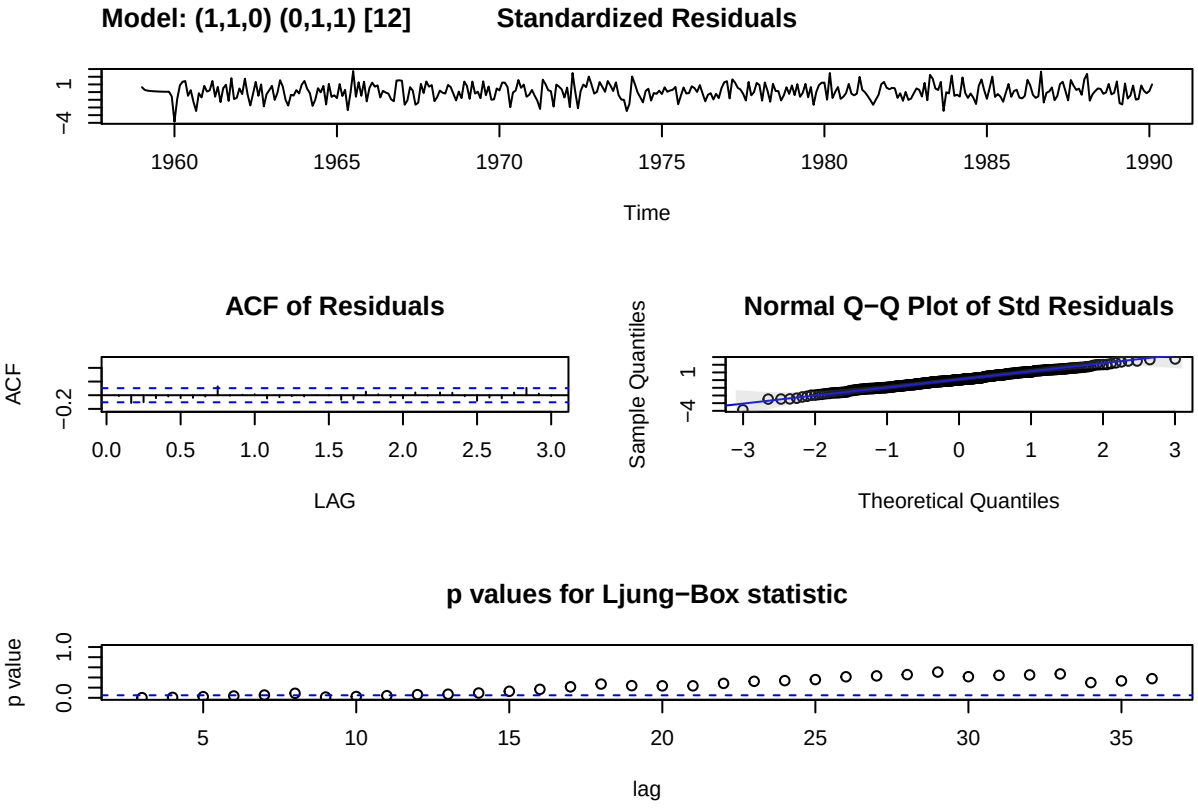


```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ma1     sma1
##       -0.3643  -0.8565
## s.e.   0.0556   0.0324
##
## sigma^2 estimated as 0.0803:  log likelihood = -65.03,  aic = 136.07
##
## $degrees_of_freedom
## [1] 359
##
## $ttable
##      Estimate     SE  t.value p.value
## ma1   -0.3643 0.0556  -6.5489       0
## sma1  -0.8565 0.0324 -26.4695       0
##
## $AIC
## [1] -1.511233
##
## $AICc
## [1] -1.505712
##
## $BIC
## [1] -2.490248
```

```r
sarima(train, 1, 1, 0, 0, 1, 1, 12) #out
```

```
## initial  value -0.935476
## iter   2 value -1.143527
## iter   3 value -1.174323
## iter   4 value -1.196953
## iter   5 value -1.203618
## iter   6 value -1.204087
## iter   7 value -1.206379
## iter   8 value -1.206518
## iter   9 value -1.206527
## iter  10 value -1.206527
## iter  10 value -1.206527
## iter  10 value -1.206527
## final  value -1.206527
## converged
## initial  value -1.223491
## iter   2 value -1.225591
## iter   3 value -1.230445
## iter   4 value -1.231046
## iter   5 value -1.231070
## iter   6 value -1.231070
## iter   6 value -1.231070
## final  value -1.231070
```

```
## converged
```

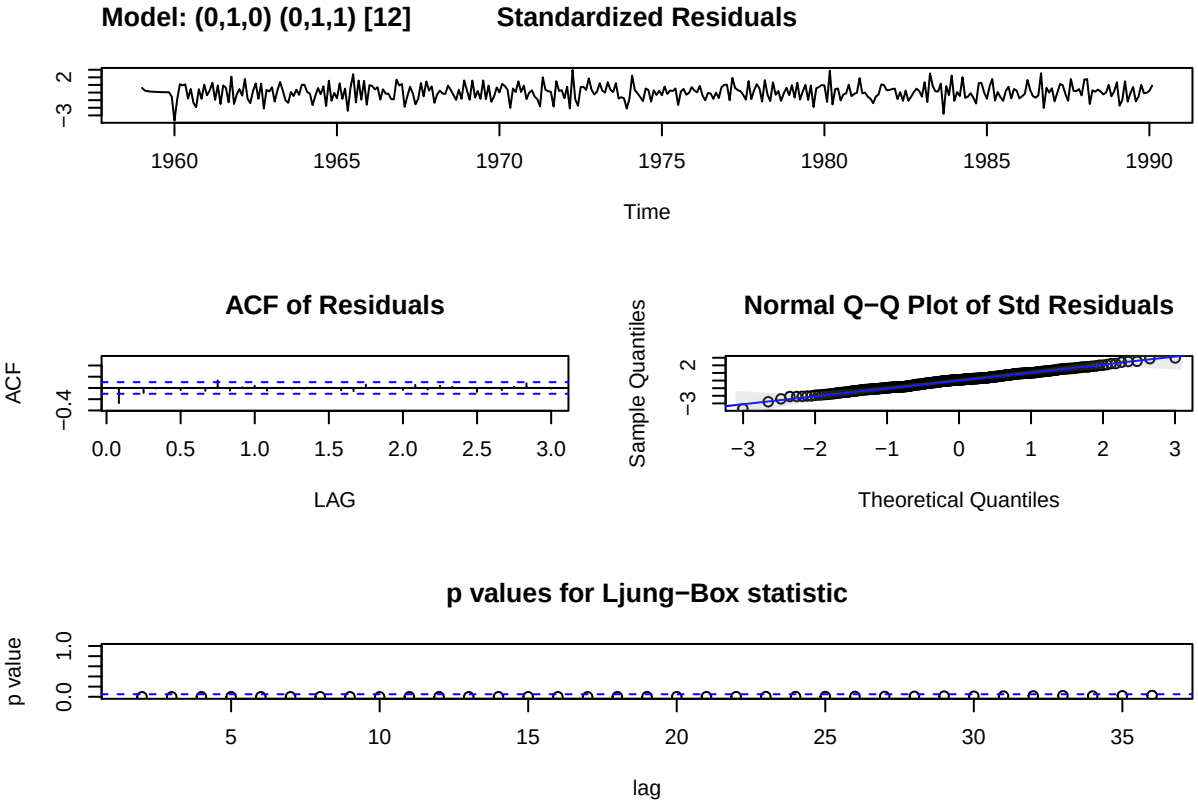



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     sma1
##       -0.3005  -0.8694
## s.e.   0.0507   0.0325
##
## sigma^2 estimated as 0.08133:  log likelihood = -67.82,  aic = 141.64
##
## $degrees_of_freedom
## [1] 359
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.3005 0.0507  -5.9255       0
## sma1  -0.8694 0.0325 -26.7575       0
##
## $AIC
## [1] -1.498573
##
## $AICc
## [1] -1.493052
```

```
##
## $BIC
## [1] -2.477588
```

```
sarima(train, 0, 1, 0, 0, 1, 1, 12) #out
```

```
## initial  value -0.935696
## iter   2 value -1.092525
## iter   3 value -1.124675
## iter   4 value -1.144499
## iter   5 value -1.149767
## iter   6 value -1.152270
## iter   7 value -1.153464
## iter   8 value -1.153585
## iter   9 value -1.153632
## iter  10 value -1.153632
## iter  10 value -1.153632
## final  value -1.153632
## converged
## initial  value -1.173470
## iter   2 value -1.181588
## iter   3 value -1.184837
## iter   4 value -1.184975
## iter   5 value -1.184979
## iter   6 value -1.184980
## iter   6 value -1.184980
## iter   6 value -1.184980
## final  value -1.184980
## converged
```

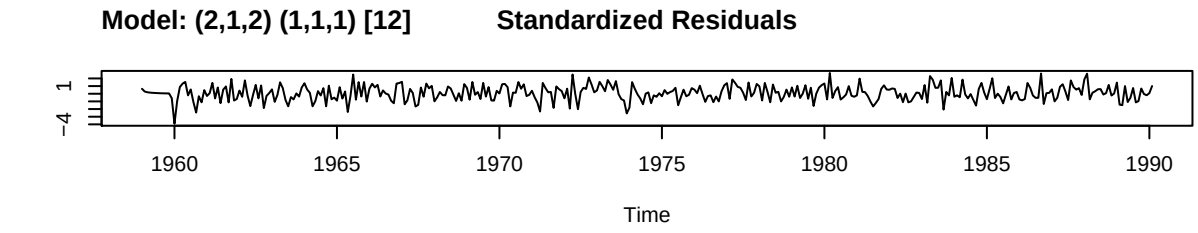


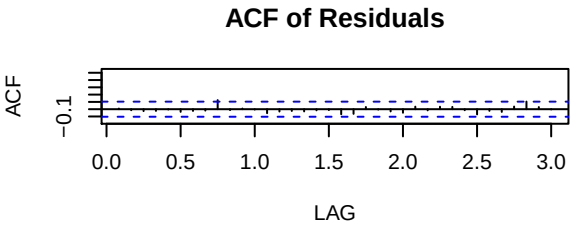
```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          sma1
##       -0.9011
## s.e.   0.0332
##
## sigma^2 estimated as 0.08844:  log likelihood = -84.46,  aic = 172.92
##
## $degrees_of_freedom
## [1] 360
##
## $ttable
##      Estimate     SE t.value p.value
## sma1  -0.9011 0.0332 -27.164       0
##
## $AIC
## [1] -1.420128
##
## $AICc
## [1] -1.414694
##
## $BIC
## [1] -2.409635
```

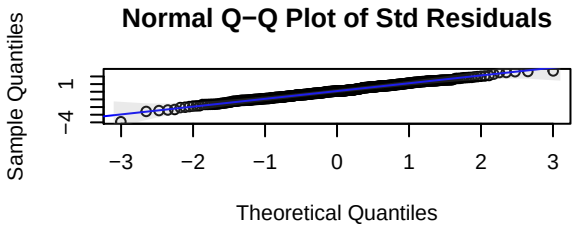
```
sarima(train, 2, 1, 2, 1, 1, 1, 12) #out
```

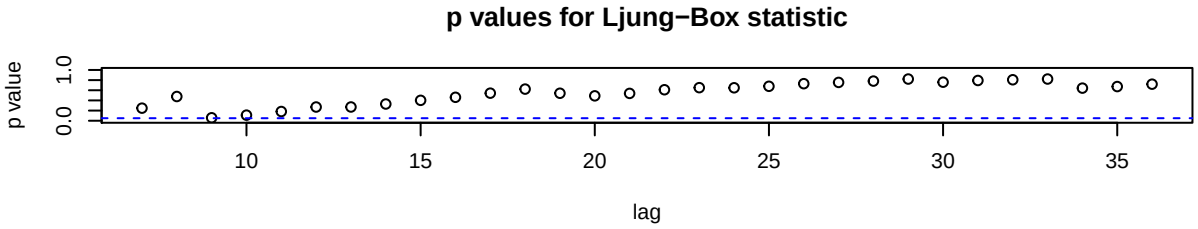
```
## initial  value -0.935325
## iter   2 value -1.127388
## iter   3 value -1.198008
## iter   4 value -1.228518
## iter   5 value -1.233525
## iter   6 value -1.234081
## iter   7 value -1.236447
## iter   8 value -1.236960
## iter   9 value -1.237198
## iter  10 value -1.237439
## iter  11 value -1.238180
## iter  12 value -1.238388
## iter  13 value -1.238509
## iter  14 value -1.238566
## iter  15 value -1.238595
## iter  16 value -1.238659
## iter  17 value -1.238837
## iter  18 value -1.238995
## iter  19 value -1.239138
## iter  20 value -1.239189
## iter  21 value -1.239235
## iter  22 value -1.239325
## iter  23 value -1.239403
```

```
## iter  24 value -1.239446
## iter  25 value -1.239454
## iter  26 value -1.239455
## iter  26 value -1.239455
## iter  26 value -1.239455
## final  value -1.239455
## converged
## initial  value -1.237923
## iter   2 value -1.241632
## iter   3 value -1.243839
## iter   4 value -1.243977
## iter   5 value -1.244079
## iter   6 value -1.244403
## iter   7 value -1.244494
## iter   8 value -1.244548
## iter   9 value -1.244575
## iter  10 value -1.244655
## iter  11 value -1.244850
## iter  12 value -1.245038
## iter  13 value -1.245322
## iter  14 value -1.245526
## iter  15 value -1.245538
## iter  16 value -1.245545
## iter  17 value -1.245550
## iter  18 value -1.245553
## iter  19 value -1.245557
## iter  20 value -1.245562
## iter  21 value -1.245563
## iter  22 value -1.245564
## iter  23 value -1.245567
## iter  24 value -1.245574
## iter  25 value -1.245585
## iter  26 value -1.245595
## iter  27 value -1.245598
## iter  28 value -1.245599
## iter  29 value -1.245599
## iter  30 value -1.245599
## iter  31 value -1.245599
## iter  32 value -1.245599
## iter  33 value -1.245599
## iter  33 value -1.245599
## iter  33 value -1.245599
## final  value -1.245599
## converged
```









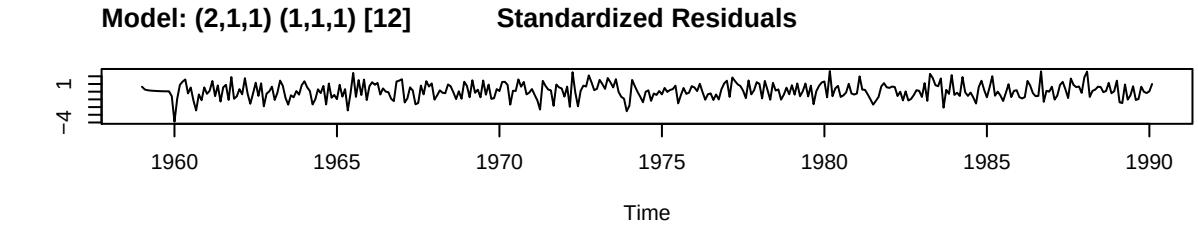

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2      ma1      ma2    sar1     sma1
##       -0.0384  0.2601  -0.3140  -0.2915  0.0181  -0.8600
## s.e.   0.4116  0.1361   0.4131   0.2427  0.0627   0.0366
##
## sigma^2 estimated as 0.07922:  log likelihood = -62.58,  aic = 139.15
##
## $degrees_of_freedom
## [1] 355
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.0384 0.4116  -0.0932  0.9258
## ar2    0.2601 0.1361   1.9107  0.0569
## ma1   -0.3140 0.4131  -0.7601  0.4477
## ma2   -0.2915 0.2427  -1.2014  0.2304
## sar1   0.0181 0.0627   0.2882  0.7733
## sma1  -0.8600 0.0366 -23.5174  0.0000
##
## $AIC
## [1] -1.503428
```

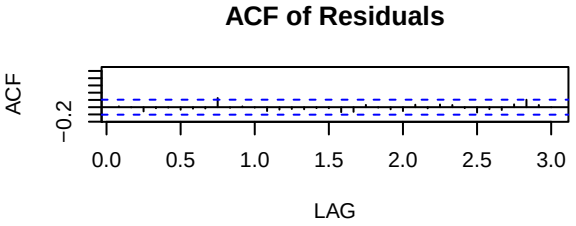
```
##
## $AICc
## [1] -1.497262
##
## $BIC
## [1] -2.440472
```

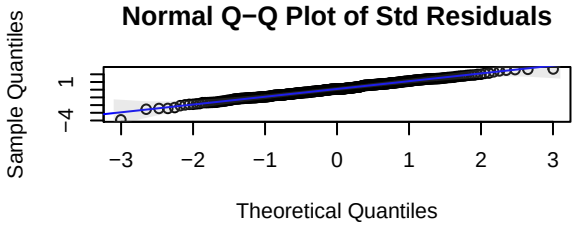
```
sarima(train, 2, 1, 1, 1, 1, 1, 12) #out
```

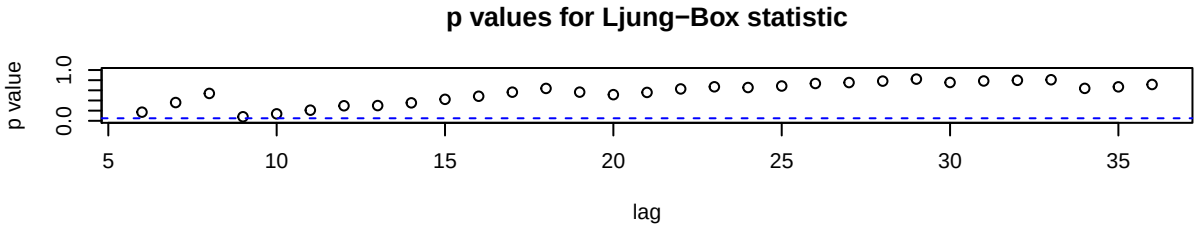
```
## initial  value -0.935325
## iter   2 value -1.137386
## iter   3 value -1.200004
## iter   4 value -1.231023
## iter   5 value -1.232466
## iter   6 value -1.234295
## iter   7 value -1.234426
## iter   8 value -1.234432
## iter   9 value -1.234435
## iter  10 value -1.234451
## iter  11 value -1.234474
## iter  12 value -1.234503
## iter  13 value -1.234509
## iter  14 value -1.234515
## iter  15 value -1.234515
## iter  16 value -1.234515
## iter  17 value -1.234516
## iter  18 value -1.234518
## iter  19 value -1.234519
## iter  20 value -1.234519
## iter  21 value -1.234519
## iter  21 value -1.234519
## iter  21 value -1.234519
## final  value -1.234519
## converged
## initial  value -1.232722
## iter   2 value -1.235874
## iter   3 value -1.238435
## iter   4 value -1.238628
## iter   5 value -1.238844
## iter   6 value -1.239849
## iter   7 value -1.240347
## iter   8 value -1.241007
## iter   9 value -1.241576
## iter  10 value -1.242744
## iter  11 value -1.244098
## iter  12 value -1.244408
## iter  13 value -1.244418
## iter  14 value -1.244482
## iter  15 value -1.244523
## iter  16 value -1.244536
## iter  17 value -1.244543
## iter  18 value -1.244544
## iter  18 value -1.244544
## iter  18 value -1.244544
## final  value -1.244544
```

```
## converged
```









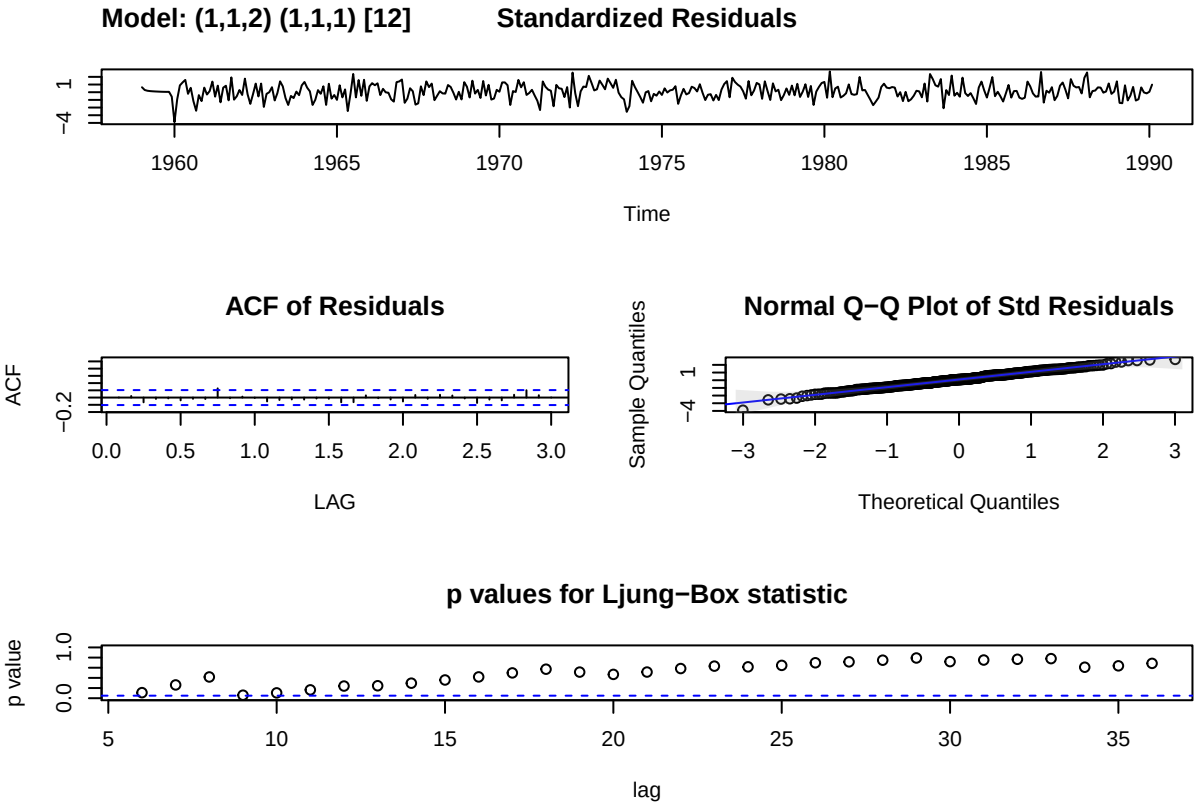

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     ar2      ma1    sar1     sma1
##       0.4015  0.1138  -0.7577  0.0132  -0.8581
## s.e.  0.1370  0.0788   0.1234  0.0624   0.0364
##
## sigma^2 estimated as 0.07939:  log likelihood = -62.96,  aic = 137.91
##
## $degrees_of_freedom
## [1] 356
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.4015 0.1370   2.9317  0.0036
## ar2    0.1138 0.0788   1.4448  0.1494
## ma1   -0.7577 0.1234  -6.1402  0.0000
## sar1   0.0132 0.0624   0.2112  0.8328
## sma1  -0.8581 0.0364 -23.5492  0.0000
##
## $AIC
## [1] -1.506587
```

```
##
## $AICc
## [1] -1.500627
##
## $BIC
## [1] -2.454124
```

```
sarima(train, 1, 1, 2, 1, 1, 1, 12) #out
```

```
## initial  value -0.936269
## iter   2 value -1.131049
## iter   3 value -1.202947
## iter   4 value -1.233329
## iter   5 value -1.235722
## iter   6 value -1.238132
## iter   7 value -1.238297
## iter   8 value -1.238354
## iter   9 value -1.238804
## iter  10 value -1.239439
## iter  11 value -1.240188
## iter  12 value -1.240467
## iter  13 value -1.240505
## iter  14 value -1.240525
## iter  15 value -1.240527
## iter  16 value -1.240527
## iter  17 value -1.240527
## iter  18 value -1.240528
## iter  19 value -1.240530
## iter  20 value -1.240531
## iter  21 value -1.240532
## iter  22 value -1.240532
## iter  22 value -1.240532
## iter  22 value -1.240532
## final  value -1.240532
## converged
## initial  value -1.237303
## iter   2 value -1.239929
## iter   3 value -1.241695
## iter   4 value -1.241887
## iter   5 value -1.241963
## iter   6 value -1.241994
## iter   7 value -1.242070
## iter   8 value -1.242309
## iter   9 value -1.242673
## iter  10 value -1.243019
## iter  11 value -1.243484
## iter  12 value -1.243526
## iter  13 value -1.243545
## iter  14 value -1.243549
## iter  15 value -1.243550
## iter  16 value -1.243552
## iter  17 value -1.243553
## iter  18 value -1.243554
## iter  19 value -1.243557
## iter  20 value -1.243563
```

```
## iter  21 value -1.243569
## iter  22 value -1.243571
## iter  23 value -1.243571
## iter  23 value -1.243571
## final  value -1.243571
## converged
```

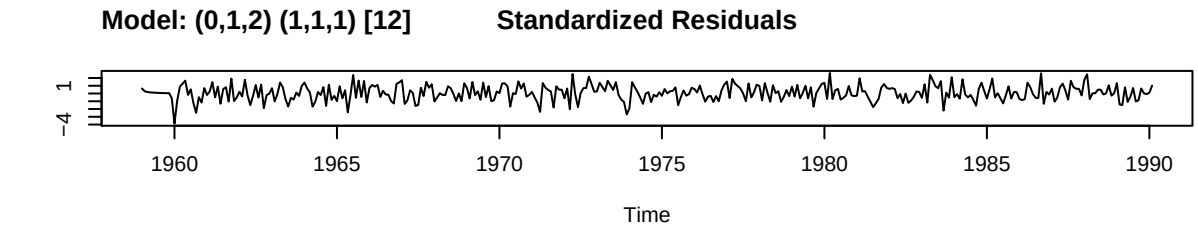


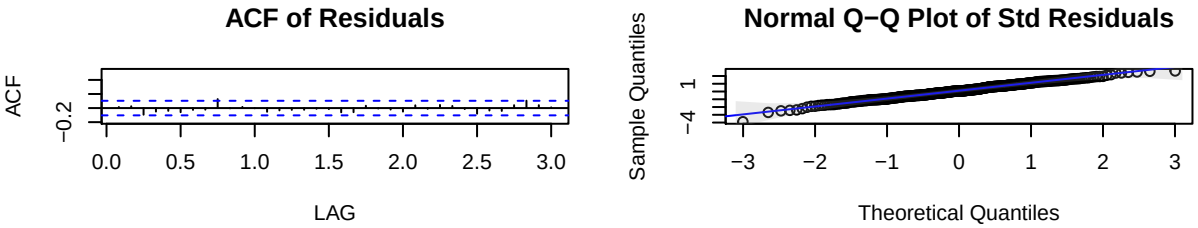

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##         ar1      ma1     ma2    sar1     sma1
##       0.620  -0.9681  0.1590  0.0108  -0.8572
## s.e.  0.264   0.2710  0.1273  0.0624   0.0365
##
## sigma^2 estimated as 0.07955:  log likelihood = -63.31,  aic = 138.62
##
## $degrees_of_freedom
## [1] 356
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.6200 0.2640   2.3488  0.0194
## ma1   -0.9681 0.2710  -3.5727  0.0004
## ma2    0.1590 0.1273   1.2485  0.2127
```

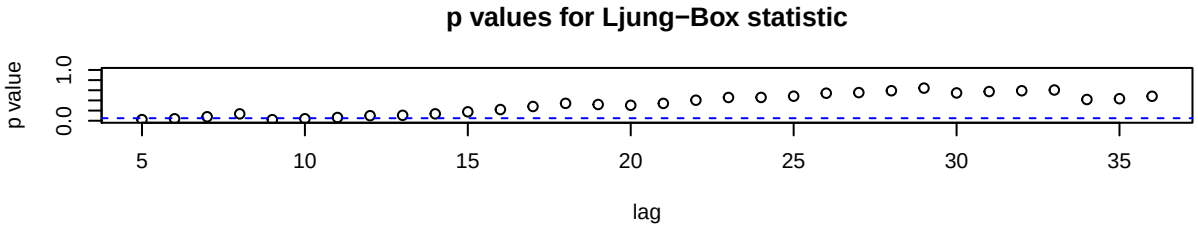
```
## sar1   0.0108 0.0624   0.1732  0.8626
## sma1  -0.8572 0.0365 -23.5082  0.0000
##
## $AIC
## [1] -1.504579
##
## $AICc
## [1] -1.498619
##
## $BIC
## [1] -2.452115
```

```
sarima(train, 0, 1, 2, 1, 1, 1, 12) #out
```

```
## initial  value -0.937023
## iter   2 value -1.160537
## iter   3 value -1.208720
## iter   4 value -1.234613
## iter   5 value -1.237611
## iter   6 value -1.238612
## iter   7 value -1.239014
## iter   8 value -1.239092
## iter   9 value -1.239095
## iter  10 value -1.239095
## iter  10 value -1.239095
## final  value -1.239095
## converged
## initial  value -1.235311
## iter   2 value -1.237901
## iter   3 value -1.240100
## iter   4 value -1.240243
## iter   5 value -1.240247
## iter   6 value -1.240247
## iter   6 value -1.240247
## iter   6 value -1.240247
## final  value -1.240247
## converged
```





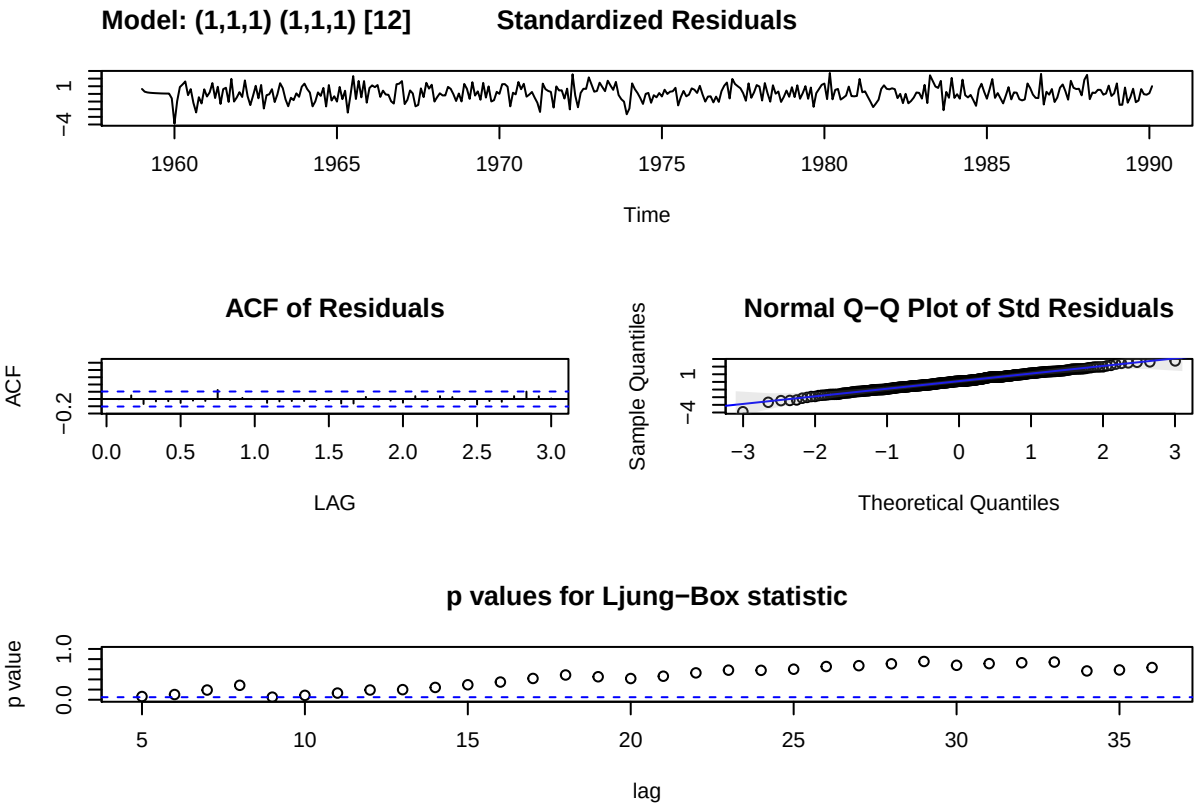



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ma1      ma2    sar1     sma1
##       -0.3563  -0.0542  0.0066  -0.8561
## s.e.   0.0531   0.0531  0.0624   0.0367
##
## sigma^2 estimated as 0.0801:  log likelihood = -64.51,  aic = 139.02
##
## $degrees_of_freedom
## [1] 357
##
## $ttable
##      Estimate     SE  t.value p.value
## ma1   -0.3563 0.0531  -6.7115  0.0000
## ma2   -0.0542 0.0531  -1.0204  0.3082
## sar1   0.0066 0.0624   0.1057  0.9159
## sma1  -0.8561 0.0367 -23.3301  0.0000
##
## $AIC
## [1] -1.503052
##
## $AICc
```

```
## [1] -1.497268
##
## $BIC
## [1] -2.461081
```

```
sarima(train, 1, 1, 1, 1, 1, 1, 12) #out
```

```
## initial  value -0.936269
## iter   2 value -1.139521
## iter   3 value -1.196132
## iter   4 value -1.231012
## iter   5 value -1.233838
## iter   6 value -1.234661
## iter   7 value -1.235331
## iter   8 value -1.235920
## iter   9 value -1.238828
## iter  10 value -1.239916
## iter  11 value -1.239970
## iter  12 value -1.240108
## iter  13 value -1.240420
## iter  14 value -1.240490
## iter  15 value -1.240514
## iter  16 value -1.240515
## iter  16 value -1.240515
## iter  16 value -1.240515
## final  value -1.240515
## converged
## initial  value -1.237173
## iter   2 value -1.239508
## iter   3 value -1.241575
## iter   4 value -1.241635
## iter   5 value -1.241637
## iter   6 value -1.241640
## iter   7 value -1.241641
## iter   8 value -1.241645
## iter   9 value -1.241647
## iter  10 value -1.241647
## iter  10 value -1.241647
## final  value -1.241647
## converged
```

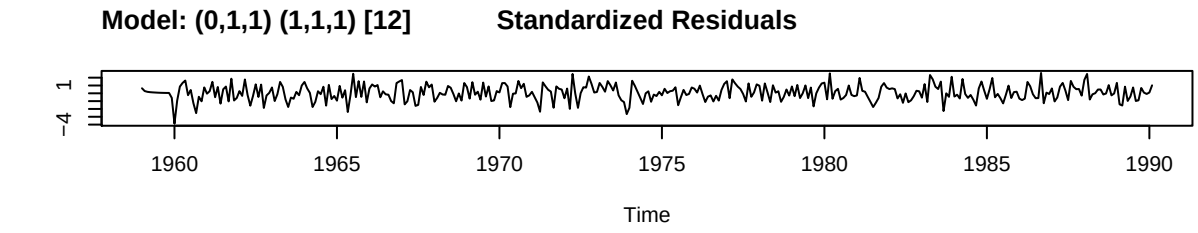


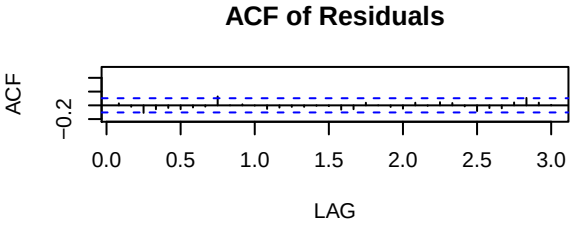

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.2522  -0.5947  0.0077  -0.8562
## s.e.  0.1504   0.1286  0.0623   0.0365
##
## sigma^2 estimated as 0.07988:  log likelihood = -64,  aic = 138
##
## $degrees_of_freedom
## [1] 357
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1    0.2522 0.1504   1.6766  0.0945
## ma1   -0.5947 0.1286  -4.6254  0.0000
## sar1   0.0077 0.0623   0.1241  0.9013
## sma1  -0.8562 0.0365 -23.4751  0.0000
##
## $AIC
## [1] -1.505876
##
## $AICc
```

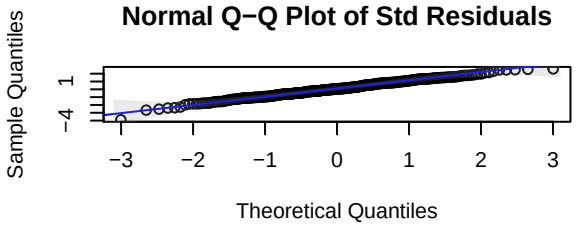
```
## [1] -1.500092
##
## $BIC
## [1] -2.463905
```

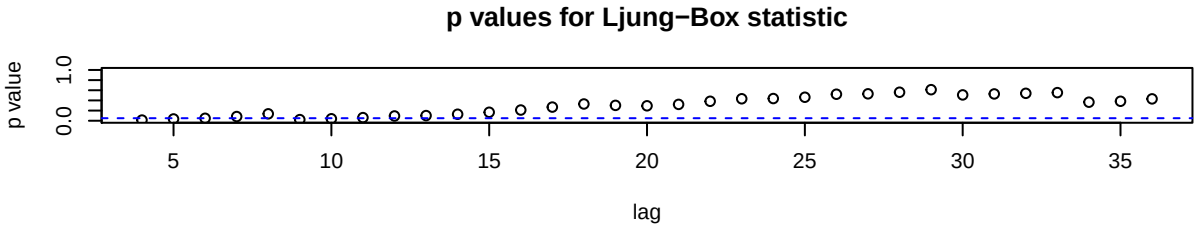
```
sarima(train, 0, 1, 1, 1, 1, 1, 12) #out
```

```
## initial  value -0.937023
## iter   2 value -1.165525
## iter   3 value -1.207601
## iter   4 value -1.220424
## iter   5 value -1.227904
## iter   6 value -1.233757
## iter   7 value -1.235128
## iter   8 value -1.235897
## iter   9 value -1.236034
## iter  10 value -1.236093
## iter  11 value -1.236094
## iter  12 value -1.236094
## iter  12 value -1.236094
## iter  12 value -1.236094
## final  value -1.236094
## converged
## initial  value -1.233581
## iter   2 value -1.236366
## iter   3 value -1.238633
## iter   4 value -1.238818
## iter   5 value -1.238821
## iter   6 value -1.238821
## iter   6 value -1.238821
## iter   6 value -1.238821
## final  value -1.238821
## converged
```







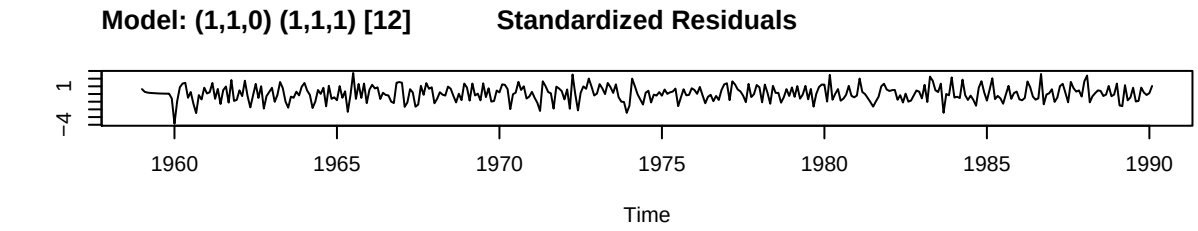


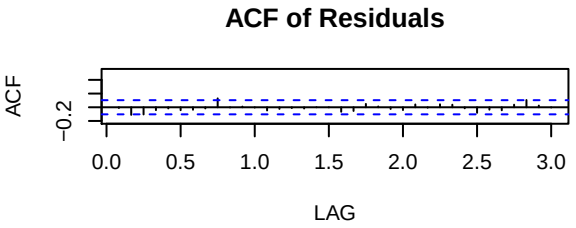

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ma1    sar1     sma1
##       -0.3632  0.0094  -0.8592
## s.e.   0.0561  0.0623   0.0367
##
## sigma^2 estimated as 0.0803:  log likelihood = -65.02,  aic = 138.04
##
## $degrees_of_freedom
## [1] 358
##
## $ttable
##      Estimate     SE  t.value p.value
## ma1   -0.3632 0.0561  -6.4699  0.0000
## sar1   0.0094 0.0623   0.1517  0.8795
## sma1  -0.8592 0.0367 -23.3969  0.0000
##
## $AIC
## [1] -1.505992
##
## $AICc
## [1] -1.500354
```

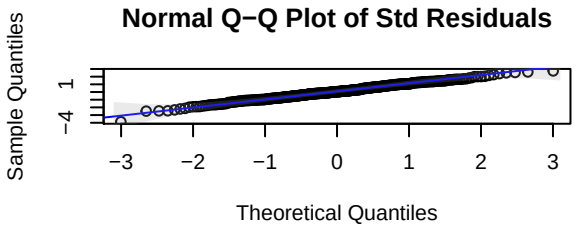
```
##
## $BIC
## [1] -2.474514
```

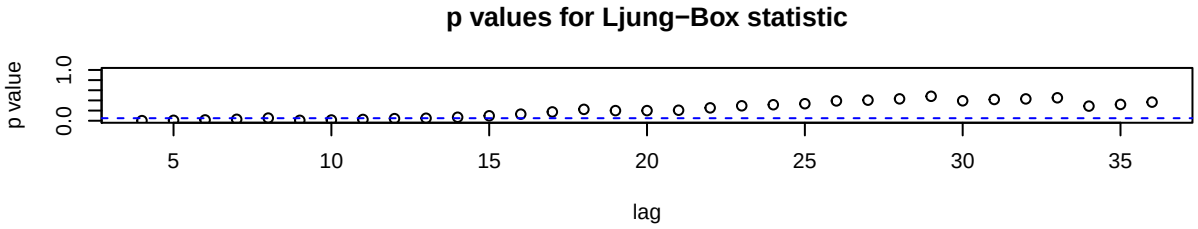
```
sarima(train, 1, 1, 0, 1, 1, 1, 12) #out
```

```
## initial  value -0.936269
## iter   2 value -1.154601
## iter   3 value -1.190101
## iter   4 value -1.211834
## iter   5 value -1.222822
## iter   6 value -1.226273
## iter   7 value -1.226484
## iter   8 value -1.226531
## iter   9 value -1.226533
## iter  10 value -1.226534
## iter  11 value -1.226535
## iter  11 value -1.226535
## iter  11 value -1.226535
## final  value -1.226535
## converged
## initial  value -1.226615
## iter   2 value -1.228854
## iter   3 value -1.231183
## iter   4 value -1.231342
## iter   5 value -1.231345
## iter   5 value -1.231345
## iter   5 value -1.231345
## final  value -1.231345
## converged
```







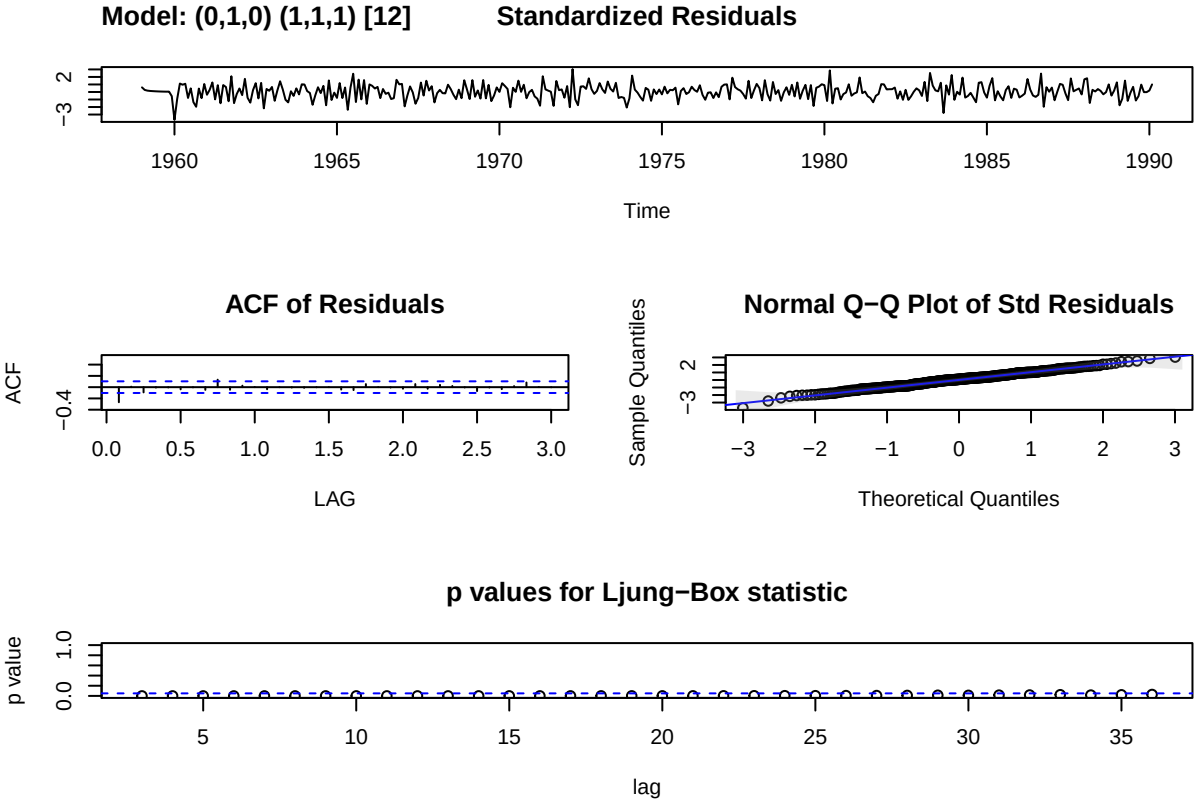



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1    sar1     sma1
##       -0.2991  0.0273  -0.8768
## s.e.   0.0508  0.0613   0.0357
##
## sigma^2 estimated as 0.08127:  log likelihood = -67.72,  aic = 143.44
##
## $degrees_of_freedom
## [1] 358
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.2991 0.0508  -5.8847  0.0000
## sar1   0.0273 0.0613   0.4452  0.6564
## sma1  -0.8768 0.0357 -24.5575  0.0000
##
## $AIC
## [1] -1.493986
##
## $AICc
## [1] -1.488349
```

```
##
## $BIC
## [1] -2.462508
```

```
sarima(train, 0, 1, 0, 1, 1, 1, 12) #out
```

```
## initial  value -0.937023
## iter   2 value -1.099323
## iter   3 value -1.135638
## iter   4 value -1.167467
## iter   5 value -1.183157
## iter   6 value -1.185617
## iter   7 value -1.185725
## iter   8 value -1.185734
## iter   9 value -1.185735
## iter   9 value -1.185735
## iter   9 value -1.185735
## final  value -1.185735
## converged
## initial  value -1.181510
## iter   2 value -1.183204
## iter   3 value -1.185867
## iter   4 value -1.185875
## iter   5 value -1.185875
## iter   5 value -1.185875
## iter   5 value -1.185875
## final  value -1.185875
## converged
```

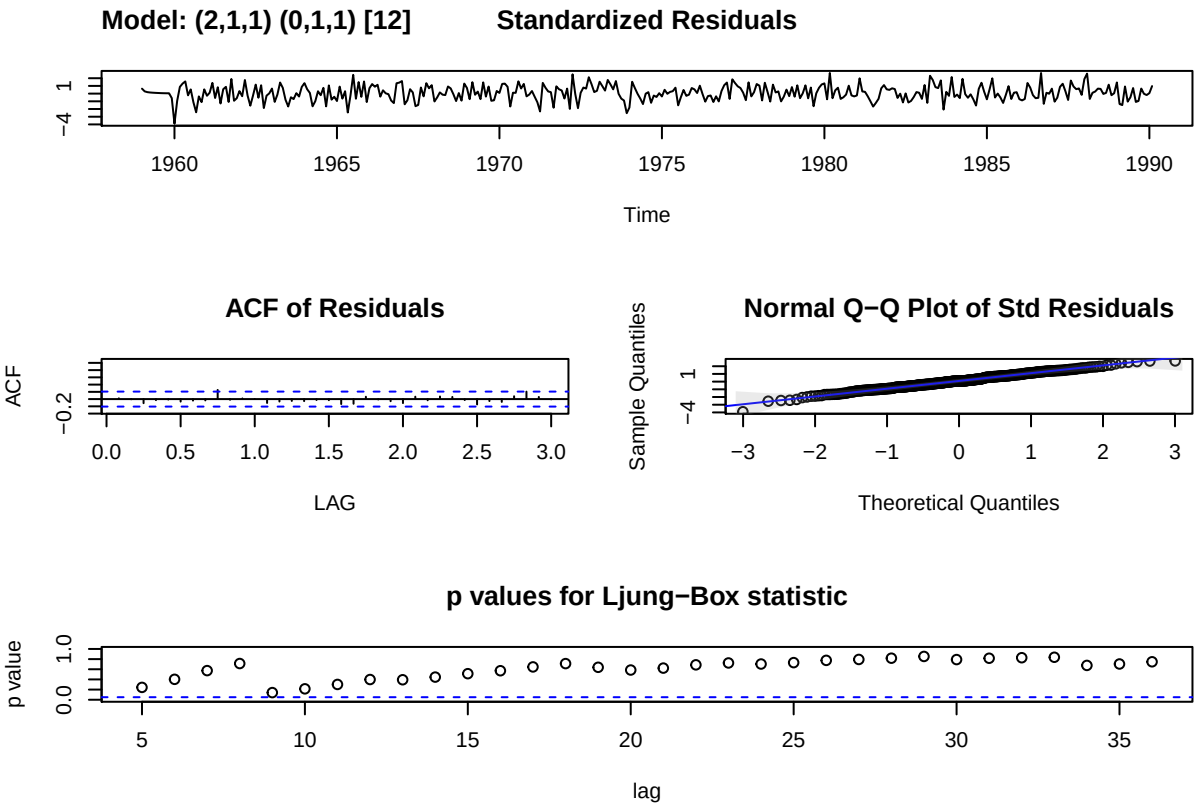


```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, optim.control = list(trace = trc,
##     REPORT = 1, reltol = tol))
##
## Coefficients:
##         sar1     sma1
##       0.0480  -0.9129
## s.e.  0.0599   0.0361
##
## sigma^2 estimated as 0.08818:  log likelihood = -84.14,  aic = 174.27
##
## $degrees_of_freedom
## [1] 359
##
## $ttable
##      Estimate     SE  t.value p.value
## sar1   0.0480 0.0599   0.8015  0.4234
## sma1  -0.9129 0.0361 -25.3121  0.0000
##
## $AIC
## [1] -1.417626
##
## $AICc
## [1] -1.412105
##
## $BIC
## [1] -2.396641
```

```
#turning.point.test(train)
#shapiro.test()
#qqnorm()




#Model parameter estimation
model1 <- sarima(train, 2, 1, 1, 0, 1, 1, 12)
```

```
## initial  value -0.935835
## iter   2 value -1.119234
## iter   3 value -1.154297
## iter   4 value -1.193102
## iter   5 value -1.207932
## iter   6 value -1.219985
## iter   7 value -1.220823
## iter   8 value -1.222039
## iter   9 value -1.222535
## iter  10 value -1.222907
## iter  11 value -1.223106
## iter  12 value -1.223593
```

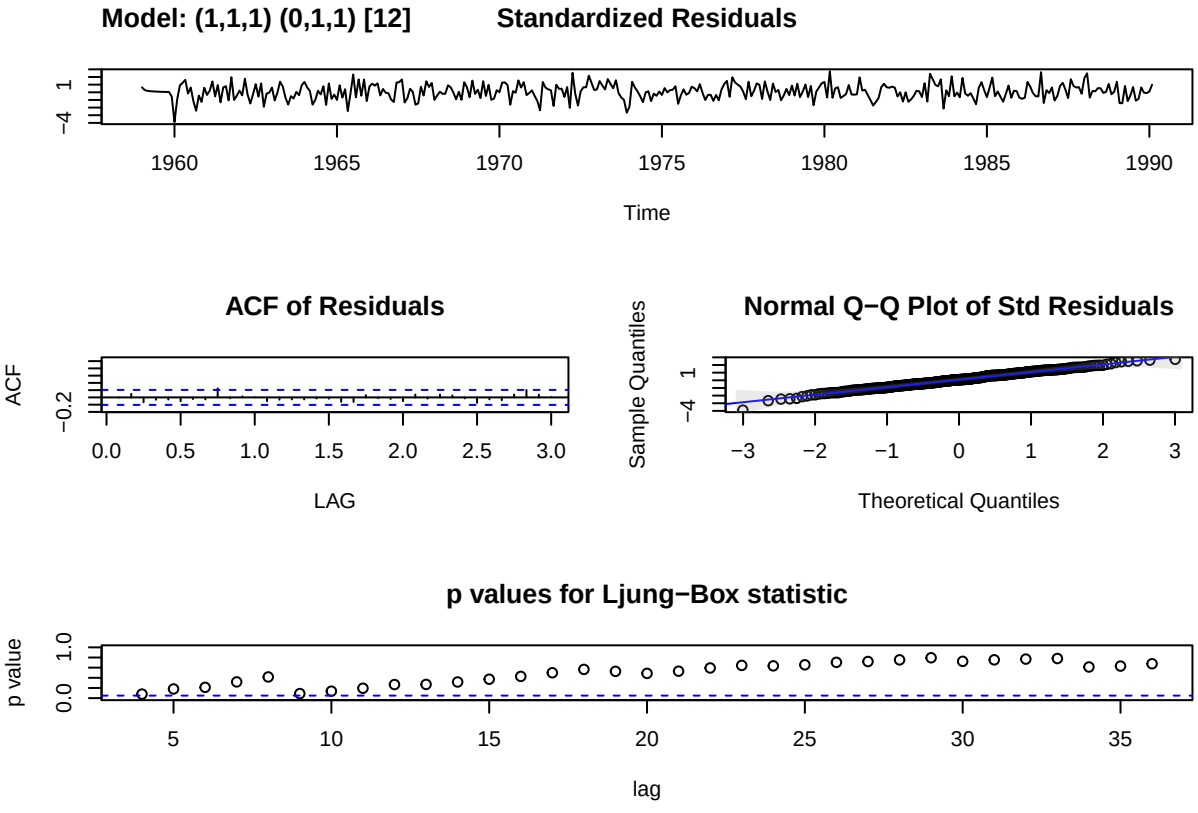
```
## iter  13 value -1.225369
## iter  14 value -1.227915
## iter  15 value -1.228375
## iter  16 value -1.229311
## iter  17 value -1.229518
## iter  18 value -1.229827
## iter  19 value -1.229880
## iter  20 value -1.229891
## iter  21 value -1.229900
## iter  22 value -1.229900
## iter  23 value -1.229900
## iter  24 value -1.229900
## iter  24 value -1.229900
## iter  24 value -1.229900
## final  value -1.229900
## converged
## initial  value -1.239498
## iter   2 value -1.242061
## iter   3 value -1.243259
## iter   4 value -1.243892
## iter   5 value -1.243949
## iter   6 value -1.244042
## iter   7 value -1.244252
## iter   8 value -1.244432
## iter   9 value -1.244479
## iter  10 value -1.244482
## iter  10 value -1.244482
## final  value -1.244482
## converged
```




```
model2 <- sarima(train, 1, 1, 1, 0, 1, 1, 12)
```

```
## initial  value -0.935476
## iter   2 value -1.116989
## iter   3 value -1.149832
## iter   4 value -1.180751
## iter   5 value -1.193873
## iter   6 value -1.205496
## iter   7 value -1.206739
## iter   8 value -1.207515
## iter   9 value -1.207585
## iter  10 value -1.207613
## iter  11 value -1.207620
## iter  12 value -1.207622
## iter  13 value -1.207622
## iter  14 value -1.207624
## iter  15 value -1.207624
## iter  16 value -1.207625
## iter  17 value -1.207625
## iter  17 value -1.207625
## iter  17 value -1.207625
## final  value -1.207625
## converged
## initial  value -1.227745
## iter   2 value -1.228668
## iter   3 value -1.235103
## iter   4 value -1.236139
## iter   5 value -1.236575
```

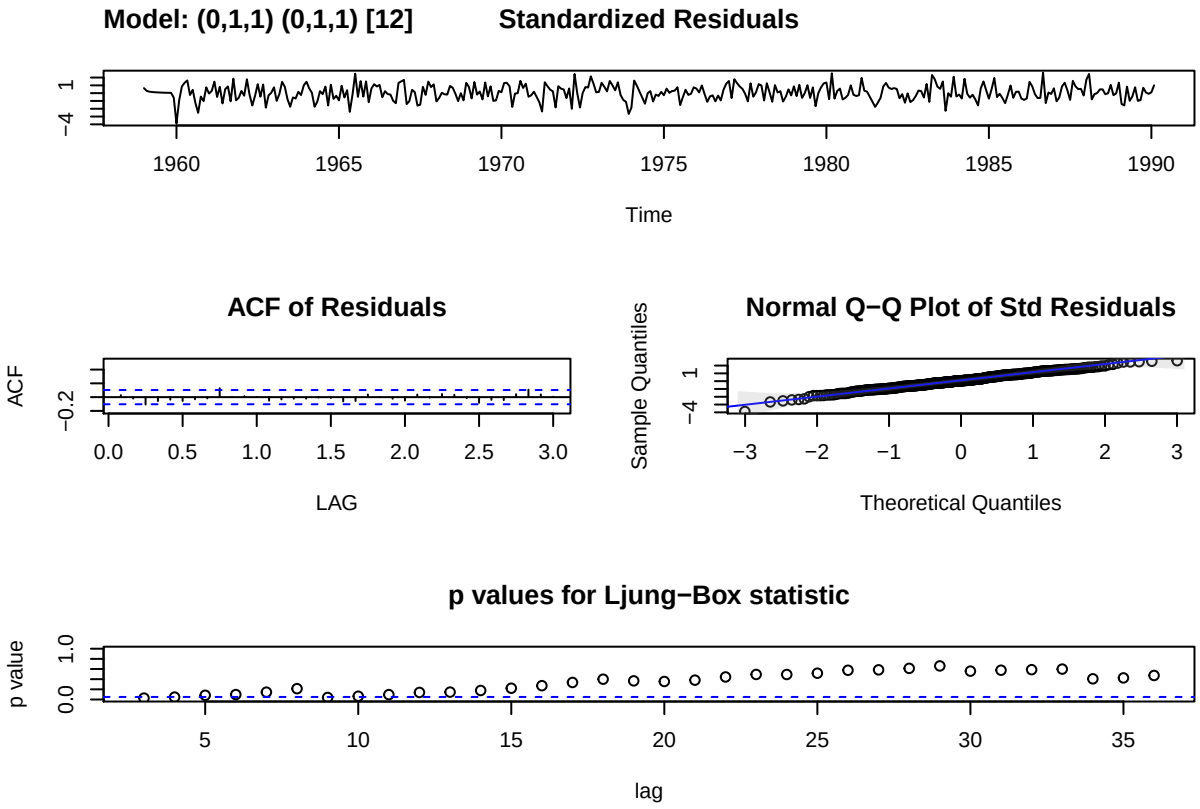
```
## iter   6 value -1.237112
## iter   7 value -1.239418
## iter   8 value -1.240757
## iter   9 value -1.241160
## iter  10 value -1.241257
## iter  11 value -1.241318
## iter  12 value -1.241447
## iter  13 value -1.241582
## iter  14 value -1.241620
## iter  15 value -1.241626
## iter  16 value -1.241626
## iter  16 value -1.241626
## iter  16 value -1.241626
## final  value -1.241626
## converged
```




```
model3 <- sarima(train, 0, 1, 1, 0, 1, 1, 12)
```

```
## initial  value -0.935696
## iter   2 value -1.149414
## iter   3 value -1.183278
## iter   4 value -1.205436
## iter   5 value -1.211805
## iter   6 value -1.212611
## iter   7 value -1.214642
## iter   8 value -1.214790
## iter   9 value -1.214798
## iter  10 value -1.214798
## iter  10 value -1.214798
```

```
## iter  10 value -1.214798
## final  value -1.214798
## converged
## initial  value -1.232451
## iter   2 value -1.235908
## iter   3 value -1.237512
## iter   4 value -1.238703
## iter   5 value -1.238787
## iter   6 value -1.238789
## iter   6 value -1.238789
## final  value -1.238789
## converged
```




```
model1$ttable[,1]
```

```
##     ar1     ar2     ma1    sma1
##  0.3994  0.1131 -0.7566 -0.8544
```

```
model2$ttable[,1]
```

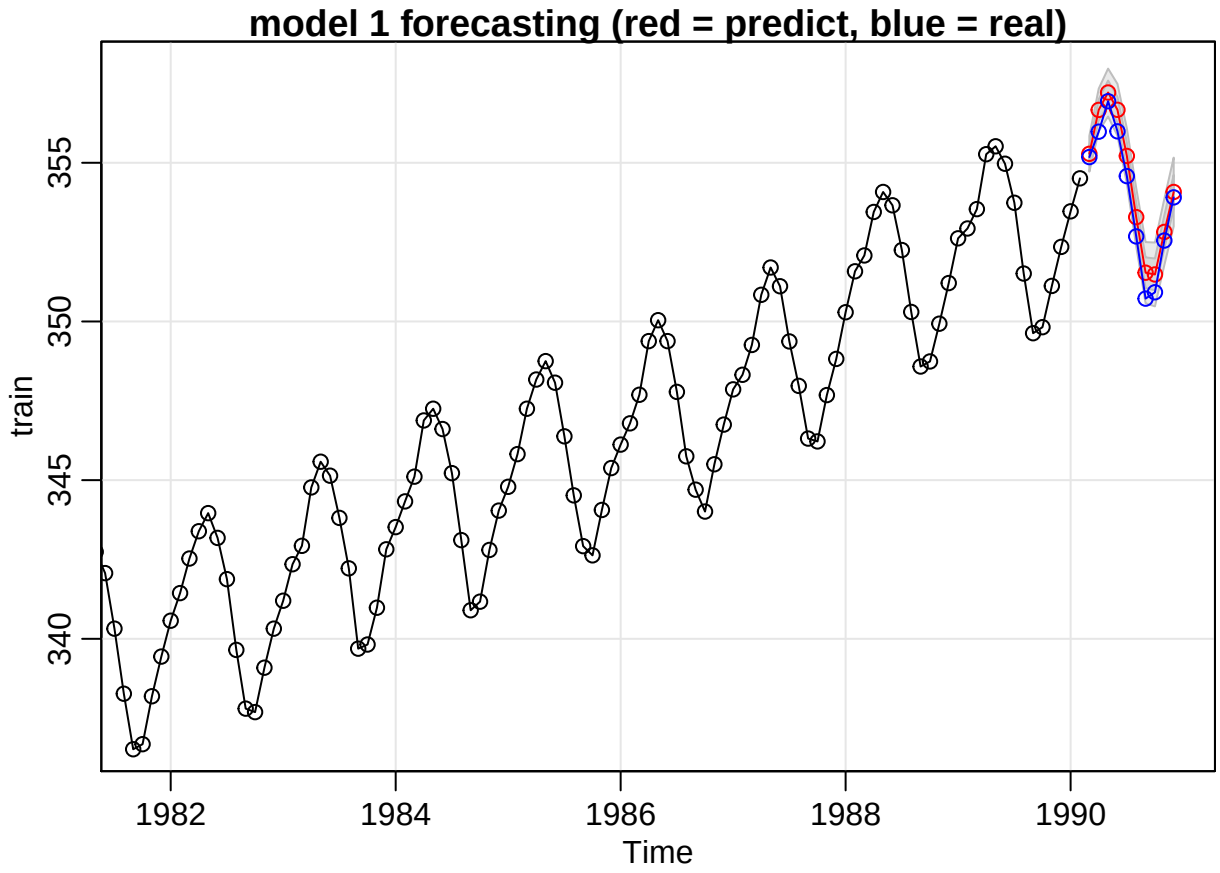
```
##     ar1     ma1    sma1
##  0.2522 -0.5953 -0.8540
```

```
model3$ttable[,1]
```

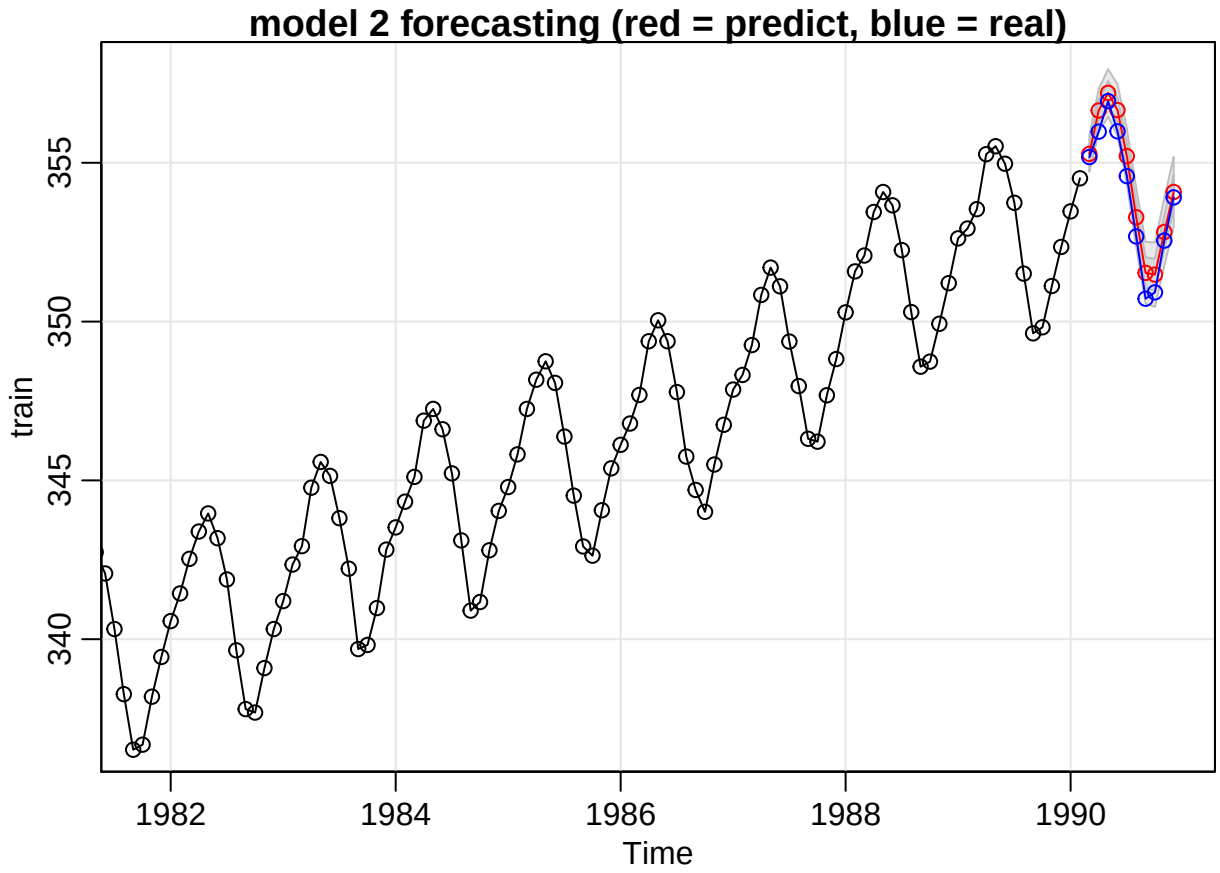
```
##     ma1    sma1
## -0.3643 -0.8565
```

```
#Forecasting
model1_for <- sarima.for(train, 10, 2, 1, 1, 0, 1, 1, 12)
lines(test, type = "o", col = "blue") #real
```

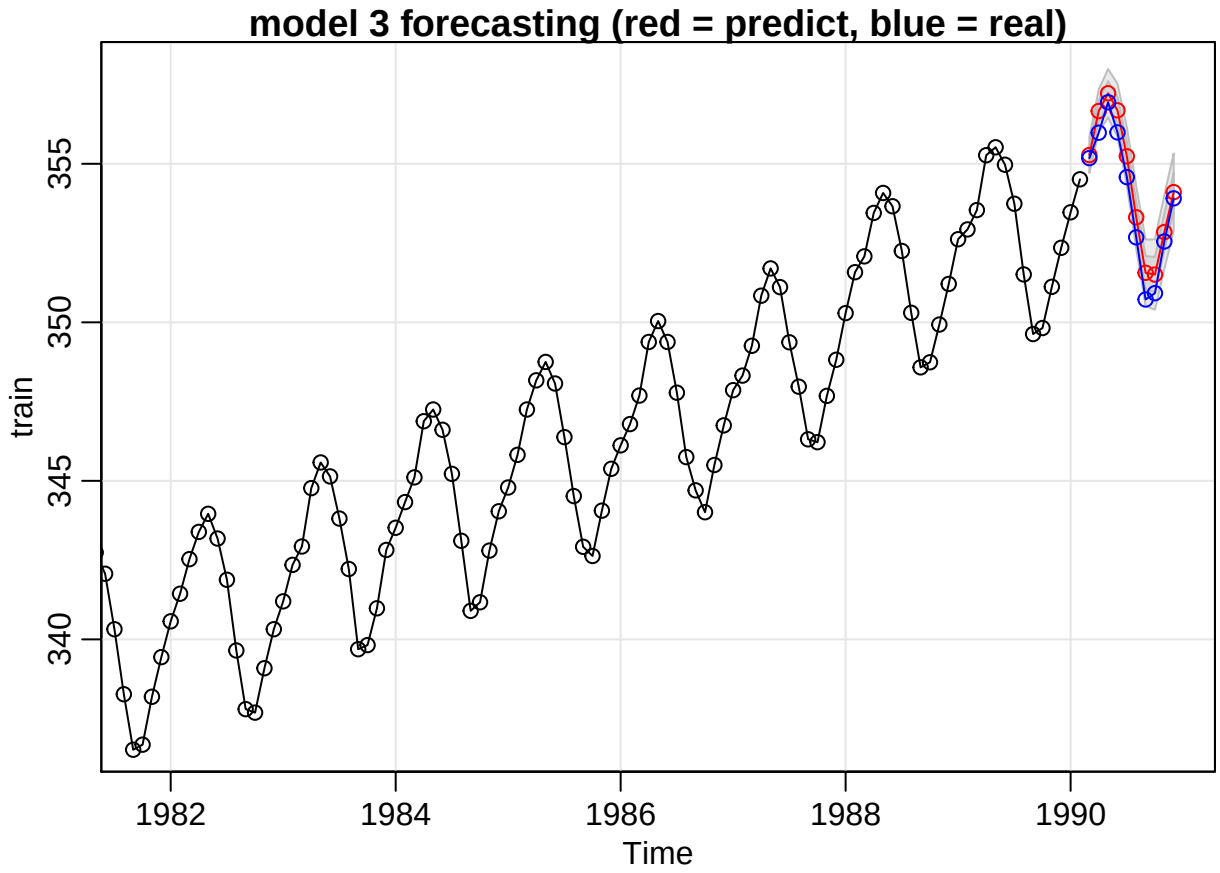
```r
title(main = "model 1 forecasting (red = predict, blue = real)")
```




```r
model2_for <- sarima.for(train, 10, 1, 1, 1, 0, 1, 1, 12)
lines(test, type = "o", col = "blue") #real
title("model 2 forecasting (red = predict, blue = real)")
```




```
model3_for <- sarima.for(train, 10, 0, 1, 1, 0, 1, 1, 12)
lines(test, type = "o", col = "blue") #real
title("model 3 forecasting (red = predict, blue = real)")
```




```
#Prediction and Standard Error table
predict_data <- data.frame(model1_prediction = model1_for$pred,
                     model1_sd = model1_for$se,
                     model2_prediction = model2_for$pred,
                     model2_sd = model2_for$se,
                     model3_prediction = model3_for$pred,
                     model3_sd = model1_for$se)
predict_data <- round(predict_data, 3)
predict_data
```

```
##    model1_prediction model1_sd model2_prediction model2_sd
## 1            355.283     0.282           355.279     0.283
## 2            356.666     0.335           356.645     0.338
## 3            357.214     0.377           357.201     0.375
## 4            356.669     0.409           356.660     0.405
## 5            355.215     0.436           355.210     0.433
## 6            353.288     0.460           353.285     0.460
## 7            351.537     0.482           351.535     0.485
## 8            351.481     0.503           351.480     0.508
## 9            352.821     0.522           352.821     0.531
## 10           354.080     0.541           354.080     0.552
##    model3_prediction model3_sd
## 1            355.275     0.282
## 2            356.663     0.335
## 3            357.225     0.377
## 4            356.686     0.409
## 5            355.237     0.436
```

```
## 6            353.312     0.460
## 7            351.563     0.482
## 8            351.507     0.503
## 9            352.847     0.522
## 10           354.106     0.541
```

```r
#MSE and Mean MSE table
mse_data <- matrix(0, 10, 3)

for(j in 1:3){
  for(i in 1:10){
    mse_data[i, j] <- (test[i] - predict_data[i, 2*j - 1])^2
  }
}
mse_data <- rbind(mse_data, colMeans(mse_data))
colnames(mse_data) <- c("Model1", "Model2", "Model3")
name <- 0
for(i in 1:nrow(mse_data) - 1){
  name[i] <- paste0("MSE: 1990, ", i+2)
}
name <- c(name, "Mean MSE")
rownames(mse_data) <- name

mse_data
```

```
##                   Model1    Model2    Model3
## MSE: 1990, 3  0.0106090 0.0098010 0.0090250
## MSE: 1990, 4  0.4705960 0.4422250 0.4664890
## MSE: 1990, 5  0.0750760 0.0681210 0.0812250
## MSE: 1990, 6  0.4610410 0.4489000 0.4844160
## MSE: 1990, 7  0.4032250 0.3969000 0.4316490
## MSE: 1990, 8  0.3696640 0.3660250 0.3994240
## MSE: 1990, 9  0.6674890 0.6642250 0.7106490
## MSE: 1990, 10 0.3147210 0.3136000 0.3445690
## MSE: 1990, 11 0.0734410 0.0734410 0.0882090
## MSE: 1990, 12 0.0289000 0.0289000 0.0384160
## Mean MSE      0.2874762 0.2812138 0.3054071
```

Comment:

Before we actually building the model, we decided to split the data into two sets. As Johnny suggested in the class, we would leave 10 data out. (from original dataset, not from the dataset differencing applied since they removed some data due to differencing) We do not want to want to use entired data set to build a model, since we do not want to re-use them when we forecase, as it would be biased and might cause over-fitting.

Our model is ARIMA(p, 1, q) $\times$ $(P, 1, Q)_{12}$, as we made seasonal and 1^{st} differencings before. After, we identified the model by using ACF, PACF, and EACF built from the training set. And, we came up with a few options:

1) Seasonal Component: At the season, it seems like the ACF is cutting off a lag 1s (s = 12), while PACF is tailing off at lags 1s, 2s, 3s, ... From there, we could come up with a SMA(1) in the season where s = 12.

2) Non-Seasonal Component: For the nonseasonals, we could inspect ACF and PACF at the lower lags. Both of them are tailing off, and we could first try an p = q = 2. And, since they are not really obvious, we would rather perform model diagnostics and use AIC, BIC, AICc to find the best model fit. (which seems pretty reasonable as this is what it seems like many people and the textbook used)

For paramter estimtions, model significance, model comparions, and model diagnostics will be computed using "sarima" function. We have compared 16 different models (please refer to the codes we have made), and came up with three good models that can feit well:

Model 1: sarima(train, 2, 1, 1, 0, 1, 1, 12) #good (p-value for ar2 is high, but AIC, AICc is the lowest, one of the ACF resdidual slightly off)

Model 2: sarima(train, 1, 1, 1, 0, 1, 1, 12) #good (p-value for ar1 is high, one of the ACF resdiual slightly off, two of p-value for Ljung box off)

Model 3: sarima(train, 0, 1, 1, 0, 1, 1, 12) #good (p-value is the lowest, BIC is the lowest, one of the ACF residual slightly off, six of p-value for Ljung box off)

There is (are) one or two outliers in standardized residual plots, and residuals do not have trend. All of ACF of residuals are within the dotted line for every lag. (check ACF individually) Normality assumption for standardized residuals seem reasonable. (just a bit off at the tails) Ljung-box statistic shows the model is adequeate as there is no correlation between residuals (check ACF groupwise) So, we could say the residuals seems like they are white noise.

Now, it is time for us to talk about forecasting. As we have mentioned earlier, we leave out the last 10 observations, and built models from there. By looking at the three models' forecastings, they all look **reasonable** as the actual 10 values are within confidence interval. (grey area) However, we would compare MSE for these three models, to decide which model would be the best. (it seems pretty reasonable although this will not guarantee the chosen model would have the lowest MSE again) We made two tables to compare actual and predicted values: one is for summary of predictions and standard errors for the three models, and the other one is for summary of MSE and mean MSE for the three models. And, we found that the model 2, ARIMA(1, 1, 1) $\times$ $(0, 1, 1)_{12}$ has the lowest mean MSE.

Spectral Analysis

11. Spectral Analysis if seasonal trend is not obvious and to get frequency

- Identify key frequencies/cycles
- Regression terms cosine and sines
- Spectral densities and Periodogram (mvspec, periodogram, spec.pgram - with and without log, spec.ar, arma.spec)
- For periodogram, try different window sizes, kernels, somoothing parameters, tapers, etc and get the best looking ones
- Smoothings
- analyze the seasonal behaviors

```
#Good reference: http://www.stat.pitt.edu/stoffer/tsa4/R_toot.htm
head(newdata,20)
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov
## 1960       -0.36  0.44  0.38  0.43 -0.31  0.19 -0.53 -0.77  0.18 -0.32
## 1961 -0.14  0.27  0.23 -0.52  0.11 -0.37  0.23  0.48 -0.06
##        Dec
## 1960  0.42
## 1961
```

```
spec.pgram
```

```
## function (x, spans = NULL, kernel = NULL, taper = 0.1, pad = 0,
##     fast = TRUE, demean = FALSE, detrend = TRUE, plot = TRUE,
##     na.action = na.fail, ...)
## {
```

```
##     series <- deparse(substitute(x))
##     x <- na.action(as.ts(x))
##     xfreq <- frequency(x)
##     x <- as.matrix(x)
##     N <- N0 <- nrow(x)
##     nser <- ncol(x)
##     if (!is.null(spans))
##         kernel <- {
##             if (is.tskernel(spans))
##                 spans
##             else kernel("modified.daniell", spans%/%2)
##         }
##     if (!is.null(kernel) && !is.tskernel(kernel))
##         stop("must specify 'spans' or a valid kernel")
##     if (detrend) {
##         t <- 1L:N - (N + 1)/2
##         sumt2 <- N * (N^2 - 1)/12
##         for (i in 1L:ncol(x)) x[, i] <- x[, i] - mean(x[, i]) -
##             sum(x[, i] * t) * t/sumt2
##     }
##     else if (demean) {
##         x <- sweep(x, 2, colMeans(x), check.margin = FALSE)
##     }
##     x <- spec.taper(x, taper)
##     u2 <- (1 - (5/8) * taper * 2)
##     u4 <- (1 - (93/128) * taper * 2)
##     if (pad > 0) {
##         x <- rbind(x, matrix(0, nrow = N * pad, ncol = ncol(x)))
##         N <- nrow(x)
##     }
##     NewN <- if (fast)
##         nextn(N)
##     else N
##     x <- rbind(x, matrix(0, nrow = (NewN - N), ncol = ncol(x)))
##     N <- nrow(x)
##     Nspec <- floor(N/2)
##     freq <- seq.int(from = xfreq/N, by = xfreq/N, length.out = Nspec)
##     xfft <- mvfft(x)
##     pgram <- array(NA, dim = c(N, ncol(x), ncol(x)))
##     for (i in 1L:ncol(x)) {
##         for (j in 1L:ncol(x)) {
##             pgram[, i, j] <- xfft[, i] * Conj(xfft[, j])/(N0 *
##                 xfreq)
##             pgram[1, i, j] <- 0.5 * (pgram[2, i, j] + pgram[N,
##                 i, j])
##         }
##     }
##     if (!is.null(kernel)) {
##         for (i in 1L:ncol(x)) for (j in 1L:ncol(x)) pgram[, i,
##             j] <- kernapply(pgram[, i, j], kernel, circular = TRUE)
##         df <- df.kernel(kernel)
##         bandwidth <- bandwidth.kernel(kernel)
##     }
##     else {
```

```
##         df <- 2
##         bandwidth <- sqrt(1/12)
##     }
##     df <- df/(u4/u2^2)
##     df <- df * (N0/N)
##     bandwidth <- bandwidth * xfreq/N
##     pgram <- pgram[2:(Nspec + 1), , , drop = FALSE]
##     spec <- matrix(NA, nrow = Nspec, ncol = nser)
##     for (i in 1L:nser) spec[, i] <- Re(pgram[1L:Nspec, i, i])
##     if (nser == 1) {
##         coh <- phase <- NULL
##     }
##     else {
##         coh <- phase <- matrix(NA, nrow = Nspec, ncol = nser *
##             (nser - 1)/2)
##         for (i in 1L:(nser - 1)) {
##             for (j in (i + 1):nser) {
##                 coh[, i + (j - 1) * (j - 2)/2] <- Mod(pgram[,
##                   i, j])^2/(spec[, i] * spec[, j])
##                 phase[, i + (j - 1) * (j - 2)/2] <- Arg(pgram[,
##                   i, j])
##             }
##         }
##     }
##     for (i in 1L:nser) spec[, i] <- spec[, i]/u2
##     spec <- drop(spec)
##     spg.out <- list(freq = freq, spec = spec, coh = coh, phase = phase,
##         kernel = kernel, df = df, bandwidth = bandwidth, n.used = N,
##         orig.n = N0, series = series, snames = colnames(x), method = ifelse(!is.null(kernel),
##             "Smoothed Periodogram", "Raw Periodogram"), taper = taper,
##         pad = pad, detrend = detrend, demean = demean)
##     class(spg.out) <- "spec"
##     if (plot) {
##         plot(spg.out, ...)
##         return(invisible(spg.out))
##     }
##     else return(spg.out)
## }
## <bytecode: 0x7f8f0862f348>
## <environment: namespace:stats>
```

```
mvspec
```

```
## function (x, spans = NULL, kernel = NULL, taper = 0, pad = 0,
##     fast = TRUE, demean = FALSE, detrend = TRUE, plot = TRUE,
##     na.action = na.fail, ...)
## {
##     na.fail = stats::na.fail
##     as.ts = stats::as.ts
##     frequency = stats::frequency
##     is.tskernel = stats::is.tskernel
##     spec.taper = stats::spec.taper
##     nextn = stats::nextn
##     mvfft = stats::mvfft
##     kernapply = stats::kernapply
```

```
##     df.kernel = stats::df.kernel
##     series <- deparse(substitute(x))
##     x <- na.action(as.ts(x))
##     xfreq <- frequency(x)
##     x <- as.matrix(x)
##     N <- N0 <- nrow(x)
##     nser <- ncol(x)
##     if (!is.null(spans))
##         kernel <- {
##             if (is.tskernel(spans))
##                 spans
##             else kernel("modified.daniell", spans%/%2)
##         }
##     if (!is.null(kernel) && !is.tskernel(kernel))
##         stop("must specify 'spans' or a valid kernel")
##     if (detrend) {
##         t <- 1:N - (N + 1)/2
##         sumt2 <- N * (N^2 - 1)/12
##         for (i in 1:ncol(x)) x[, i] <- x[, i] - mean(x[, i]) -
##             sum(x[, i] * t) * t/sumt2
##     }
##     else if (demean) {
##         x <- sweep(x, 2, colMeans(x))
##     }
##     x <- spec.taper(x, taper)
##     u2 <- (1 - (5/8) * taper * 2)
##     u4 <- (1 - (93/128) * taper * 2)
##     if (pad > 0) {
##         x <- rbind(x, matrix(0, nrow = N * pad, ncol = ncol(x)))
##         N <- nrow(x)
##     }
##     NewN <- if (fast)
##         nextn(N)
##     else N
##     x <- rbind(x, matrix(0, nrow = (NewN - N), ncol = ncol(x)))
##     N <- nrow(x)
##     Nspec <- floor(N/2)
##     freq <- seq(from = xfreq/N, by = xfreq/N, length = Nspec)
##     xfft <- mvfft(x)
##     pgram <- array(NA, dim = c(N, ncol(x), ncol(x)))
##     for (i in 1:ncol(x)) {
##         for (j in 1:ncol(x)) {
##             pgram[, i, j] <- xfft[, i] * Conj(xfft[, j])/(N0 *
##                 xfreq)
##             pgram[1, i, j] <- 0.5 * (pgram[2, i, j] + pgram[N,
##                 i, j])
##         }
##     }
##     if (!is.null(kernel)) {
##         for (i in 1:ncol(x)) for (j in 1:ncol(x)) pgram[, i,
##             j] <- kernapply(pgram[, i, j], kernel, circular = TRUE)
##         df <- df.kernel(kernel)
##         Lh = 1/sum(kernel[-kernel$m:kernel$m]^2)
##     }
```

```
##     else {
##         df <- 2
##         Lh <- 1
##     }
##     df <- df/(u4/u2^2)
##     df <- df * (N0/N)
##     bandwidth <- Lh * xfreq/N
##     pgram <- pgram[2:(Nspec + 1), , , drop = FALSE]
##     spec <- matrix(NA, nrow = Nspec, ncol = nser)
##     for (i in 1:nser) spec[, i] <- Re(pgram[1:Nspec, i, i])
##     if (nser == 1) {
##         coh <- phase <- NULL
##     }
##     else {
##         coh <- phase <- matrix(NA, nrow = Nspec, ncol = nser *
##             (nser - 1)/2)
##         for (i in 1:(nser - 1)) {
##             for (j in (i + 1):nser) {
##                 coh[, i + (j - 1) * (j - 2)/2] <- Mod(pgram[,
##                   i, j])^2/(spec[, i] * spec[, j])
##                 phase[, i + (j - 1) * (j - 2)/2] <- Arg(pgram[,
##                   i, j])
##             }
##         }
##     }
##     for (i in 1:nser) spec[, i] <- spec[, i]/u2
##     spec <- drop(spec)
##     fxx = array(NA, dim = c(nser, nser, Nspec))
##     for (k in 1:Nspec) {
##         fxx[, , k] = pgram[k, , ]
##     }
##     spg.out <- list(freq = freq, spec = spec, coh = coh, phase = phase,
##         kernel = kernel, df = df, bandwidth = bandwidth, fxx = fxx,
##         Lh = Lh, n.used = N, orig.n = N0, series = series, snames = colnames(x),
##         method = ifelse(!is.null(kernel), "Smoothed Periodogram",
##             "Raw Periodogram"), taper = taper, pad = pad, detrend = detrend,
##         demean = demean)
##     class(spg.out) <- "spec"
##     if (plot) {
##         plot(spg.out, ...)
##         return(invisible(spg.out))
##     }
##     else return(spg.out)
## }
## <environment: namespace:astsa>
```

```
spec.ar
```

```
## function (x, n.freq, order = NULL, plot = TRUE, na.action = na.fail,
##     method = "yule-walker", ...)
## {
##     if (!is.list(x)) {
##         series <- deparse(substitute(x))
##         x <- na.action(as.ts(x))
##         xfreq <- frequency(x)
```

```
##         nser <- NCOL(x)
##         x <- ar(x, is.null(order), order, na.action = na.action,
##             method = method)
##     }
##     else {
##         cn <- match(c("ar", "var.pred", "order"), names(x))
##         if (anyNA(cn))
##             stop("'x' must be a time series or an ar() fit")
##         series <- x$series
##         xfreq <- x$frequency
##         if (is.array(x$ar))
##             nser <- dim(x$ar)[2L]
##         else nser <- 1
##     }
##     order <- x$order
##     if (missing(n.freq))
##         n.freq <- 500
##     freq <- seq.int(0, 0.5, length.out = n.freq)
##     if (nser == 1) {
##         coh <- phase <- NULL
##         var.p <- as.vector(x$var.pred)
##         spec <- if (order >= 1) {
##             cs <- outer(freq, 1L:order, function(x, y) cos(2 *
##                 pi * x * y)) %*% x$ar
##             sn <- outer(freq, 1L:order, function(x, y) sin(2 *
##                 pi * x * y)) %*% x$ar
##             var.p/(xfreq * ((1 - cs)^2 + sn^2))
##         }
##         else rep.int(var.p/xfreq, length(freq))
##     }
##     else .NotYetImplemented()
##     spg.out <- list(freq = freq * xfreq, spec = spec, coh = coh,
##         phase = phase, n.used = nrow(x), series = series, method = paste0("AR (",
##             order, ") spectrum "))
##     class(spg.out) <- "spec"
##     if (plot) {
##         plot(spg.out, ci = 0, ...)
##         invisible(spg.out)
##     }
##     else spg.out
## }
## <bytecode: 0x7f8f08c9fb80>
## <environment: namespace:stats>
```

```
arma.spec
```

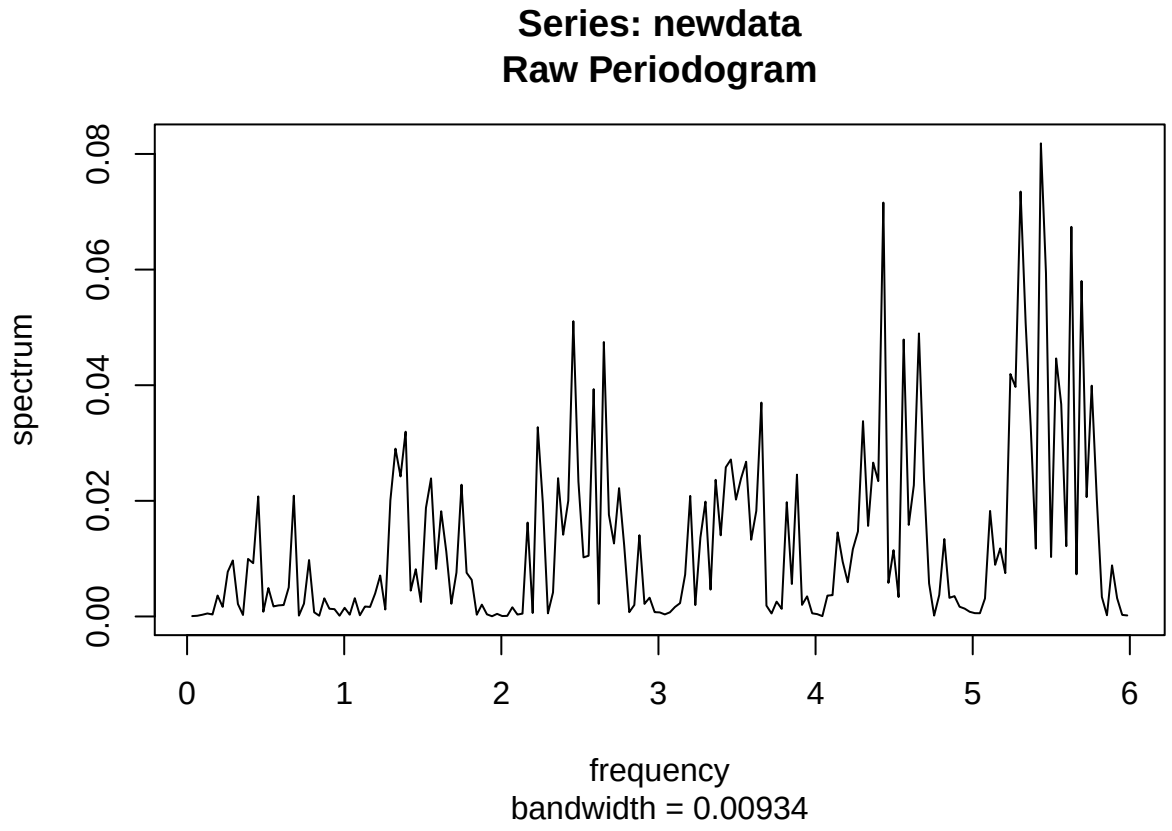
```
## function (ar = 0, ma = 0, var.noise = 1, n.freq = 500, ...)
## {
##     plot = graphics::plot
##     check <- 0
##     ar.poly <- c(1, -ar)
##     z.ar <- base::polyroot(ar.poly)
##     if (any(abs(z.ar) <= 1)) {
##         cat("WARNING: Model Not Causal", "\n")
##         check <- check + 1
```

```
##     }
##     ma.poly <- c(1, ma)
##     z.ma <- base::polyroot(ma.poly)
##     if (any(abs(z.ma) <= 1)) {
##         cat("WARNING: Model Not Invertible", "\n")
##         check <- check + 1
##     }
##     if (check > 0)
##         stop("Try Again")
##     ar.order <- length(ar)
##     ma.order <- length(ma)
##     for (i in 1:ar.order) {
##         if ((ar == 0 & ar.order == 1) || (ma == 0 & ma.order ==
##             1))
##             break
##         if (any(abs(z.ar[i] - z.ma[1:ma.order]) < 0.001)) {
##             cat("WARNING: Parameter Redundancy", "\n")
##             break
##         }
##     }
##     freq <- seq.int(0, 0.5, length.out = n.freq)
##     cs.ar <- outer(freq, 1:ar.order, function(x, y) cos(2 * pi *
##         x * y)) %*% ar
##     sn.ar <- outer(freq, 1:ar.order, function(x, y) sin(2 * pi *
##         x * y)) %*% ar
##     cs.ma <- outer(freq, 1:ma.order, function(x, y) cos(2 * pi *
##         x * y)) %*% -ma
##     sn.ma <- outer(freq, 1:ma.order, function(x, y) sin(2 * pi *
##         x * y)) %*% -ma
##     spec <- var.noise * ((1 - cs.ma)^2 + sn.ma^2)/((1 - cs.ar)^2 +
##         sn.ar^2)
##     spg.out <- list(freq = freq, spec = spec)
##     class(spg.out) <- "spec"
##     plot(spg.out, ci = 0, ...)
##     return(invisible(spg.out))
## }
## <environment: namespace:astsa>
```

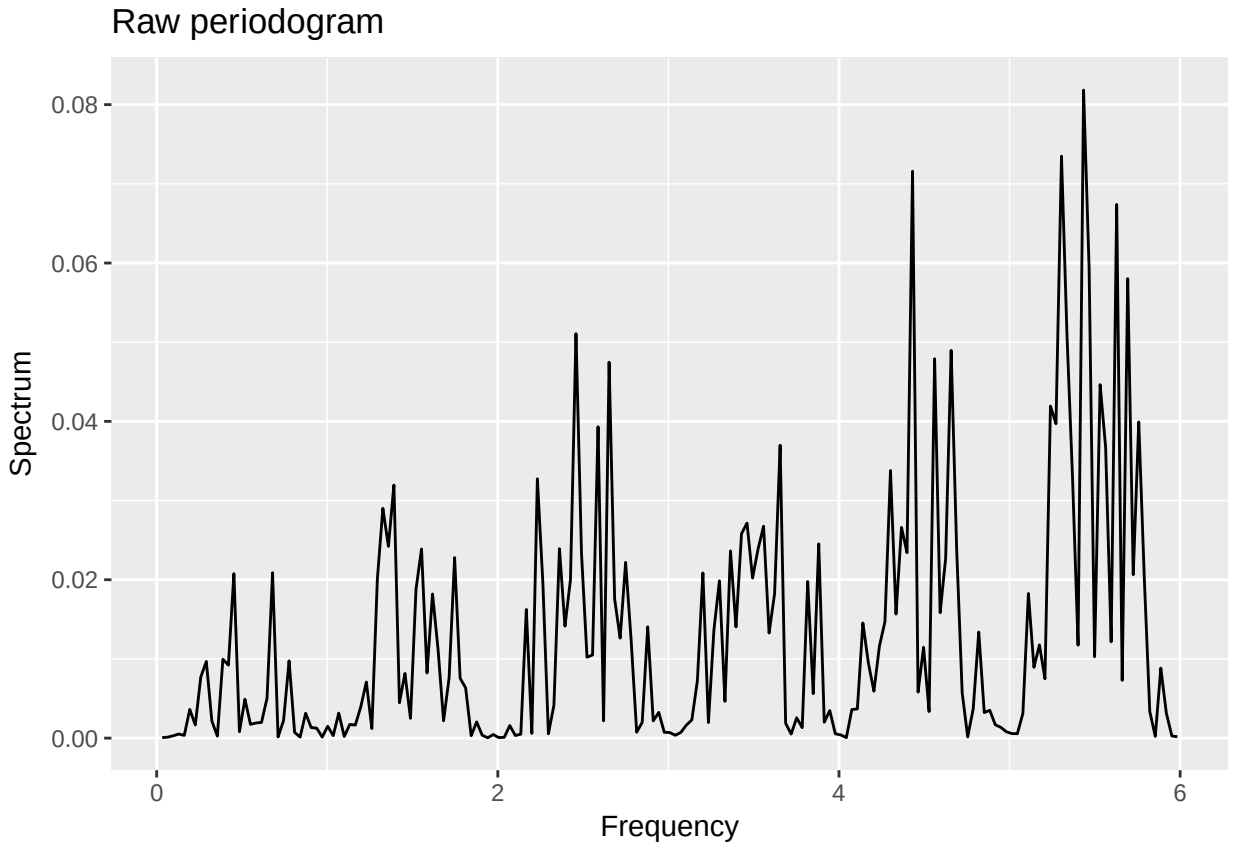
```
# #from HW
#
#
# plotPD <- function(data, smooth){
#   k <- kernel("daniell", if(smooth) floor(sqrt(length(data))) else 0)
#
#   # name of a kernel (currently "daniell", "dirichlet", "fejer" or"modified.daniell"
#
#   title <- sprintf("Raw periodogram for %d samples", length(data))
#   if(smooth){
#     title <- sprintf("Smoothed periodogram for %d samples", length(data))
#   }
#
#   p <- spec.pgram(data, k, taper = 0, log = "no", main = title)
#   df <- p$df
#   U <- df / qchisq(0.025, df)
```

```
#   L <- df / qchisq(0.975, df)
#   len <- length(p$spec)
#   idx <- round(len/5)
#   #Confidence Interval
#   c(p$spec[idx] * L, p$spec[idx] * U)
#
# }
#
# plotPD(newdata, F)
# plotPD(newdata, T)
#
#
#
# #=============================================================
#
# #outsource
#
# par(mfcol=c(2,2))
# autoplot(newdata, main="Original data")
# muspec(newdata, spans=c(5,5), plot=TRUE, taper=.1, log="no") # nonparametric spectral estimate
# spec.ar(newdata, log="no")                      # parametric spectral estimate
#
# #=============================================================
#
# del<-0.1 # sampling interval
# x.spec <- spectrum(newdata,log="no",span=10,plot=FALSE)
# spx <- x.spec$freq/del
# spy <- 2*x.spec$spec
# plot(spy~spx,xlab="frequency",ylab="spectral density",type="l")
#
#
# #=============================================================
#
# raw.spec <- spec.pgram(newdata, taper = 0)
# plot(raw.spec)
# plot(raw.spec, log = "no")

#The only way to get smooth estimates of the power spectrum is by taking moving averages of the periodo
```

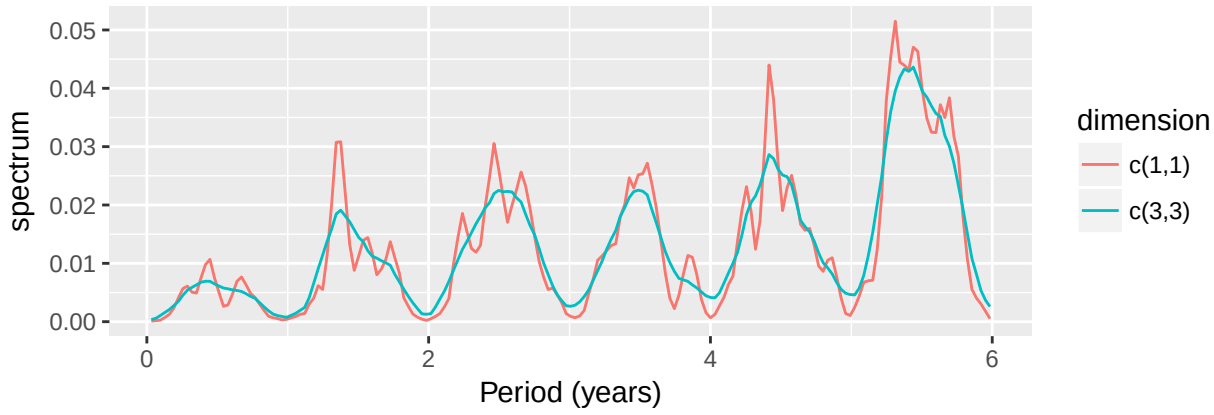
```
#=============================================================start


# Effect of different span on Kernel


k0.smooth <- spec.pgram(newdata, log='no', taper=0, pad=0, fast=FALSE, demean=TRUE, detrend=FALSE)
```

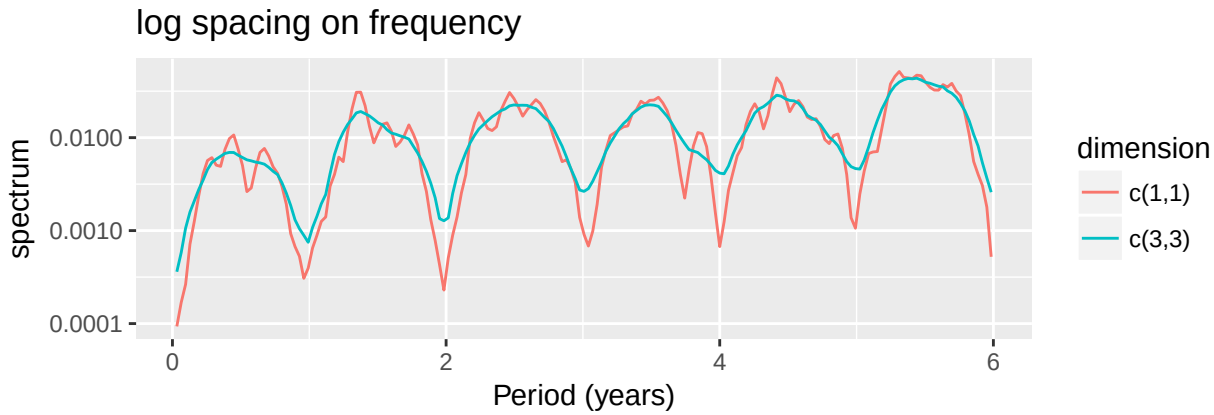



```
autoplot(k0.smooth, log = 'no', main = "Raw periodogram")
```

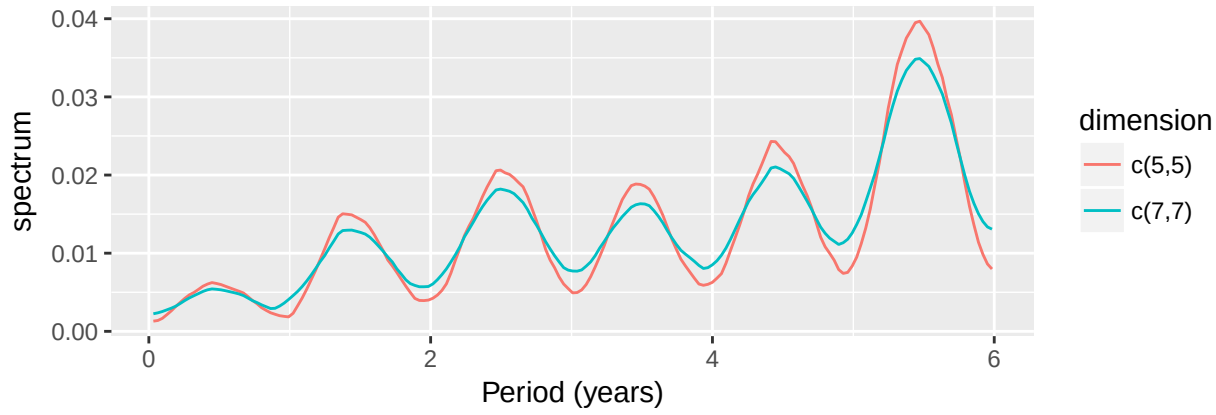



```
k1 <- kernel("daniell", c(1, 1))
k1.smooth <- spec.pgram(newdata, kernel = k1, log = 'no', taper = 0, plot = FALSE)
smooth.df <- data.frame(freq = k1.smooth$freq, `c(1,1)` = k1.smooth$spec)
names(smooth.df) <- c("frequency", "c(1,1)")
# Add other smooths
k2 <- kernel("daniell", c(3, 3))
smooth.df[, "c(3,3)"] <- spec.pgram(newdata, kernel = k2, log = 'no', taper = 0, plot = FALSE)$spec
k3 <- kernel("daniell", c(5, 5))
smooth.df[, "c(5,5)"] <- spec.pgram(newdata, kernel = k3, log = 'no', taper = 0, plot = FALSE)$spec

k4 <- kernel("daniell", c(7, 7))
smooth.df[, "c(7,7)"] <- spec.pgram(newdata, kernel = k4, log = 'no', taper = 0, plot = FALSE)$spec

# Melt dataframe in order to plot three graph together

smooth.df1 <- melt(smooth.df[,1:3], variable.name = "dimension",
                   id.vars = "frequency", value.name = "spectrum")
plot1 <- ggplot(data = subset(smooth.df1)) + geom_path(aes(x = frequency, y = spectrum, color = dimensi


plot2 <- ggplot(data = subset(smooth.df1)) +
  geom_path(aes(x = frequency, y = spectrum, color = dimension)) +
  scale_x_continuous("Period (years)") +  scale_y_log10() +
  labs(title = "log spacing on frequency")

grid.arrange(plot1, plot2)
```

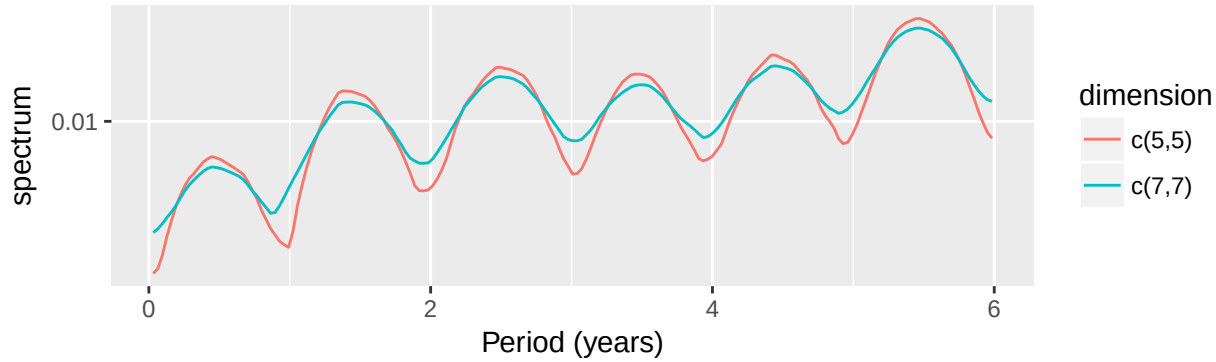





```
smooth.df2 <- melt(smooth.df[,c(1,4:5)], variable.name = "dimension",
                   id.vars = "frequency", value.name = "spectrum")
plot3 <- ggplot(data = subset(smooth.df2)) + geom_path(aes(x = frequency, y = spectrum, color = dimensi


plot4 <- ggplot(data = subset(smooth.df2)) +
  geom_path(aes(x = frequency, y = spectrum, color = dimension)) +
  scale_x_continuous("Period (years)") +  scale_y_log10() +
  labs(title = "log spacing on frequency")

grid.arrange(plot3, plot4)
```



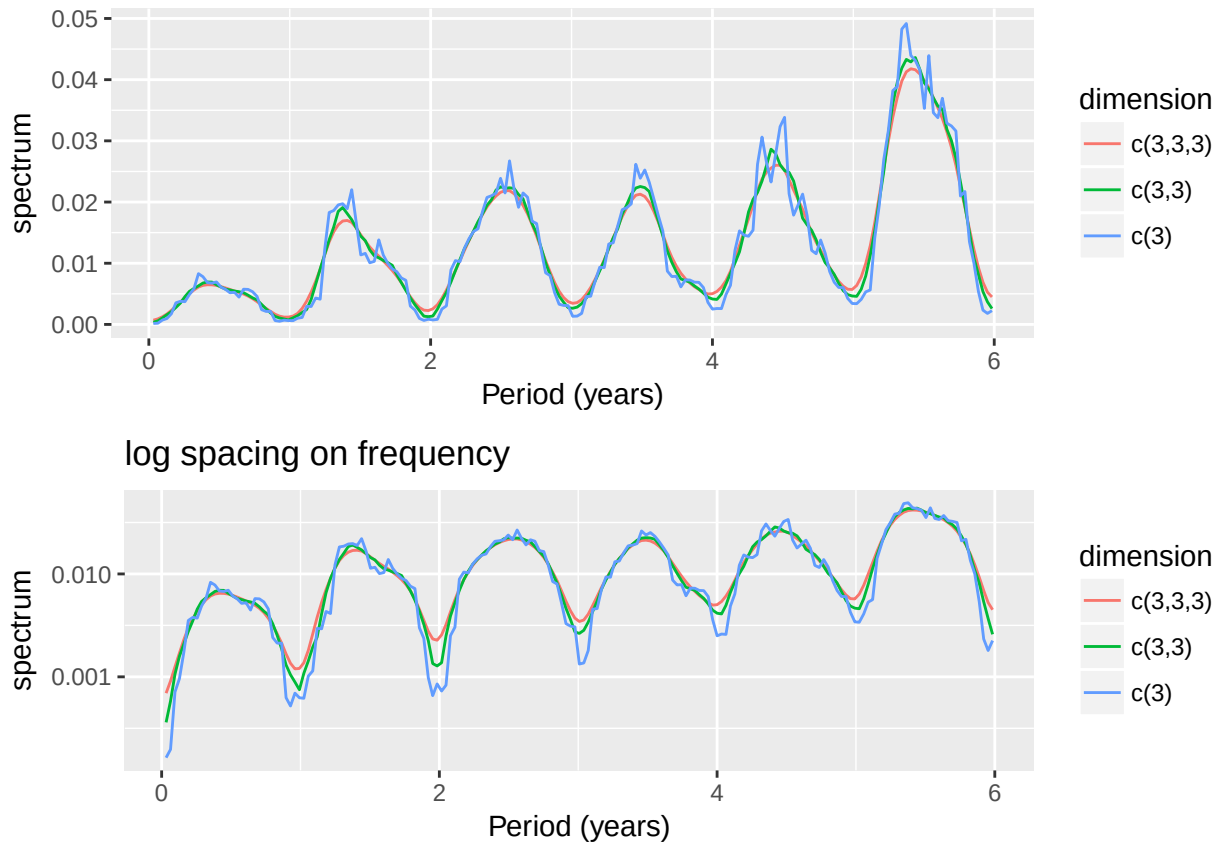
log spacing on frequency



***k4 looks smooth enough and has reasonable number of peaks

```
#=============================================================start


# Effect of different span(dimention) on Kernel

k1 <- kernel("daniell", c(3, 3, 3))
k1.smooth <- spec.pgram(newdata, kernel = k1, log = 'no', taper = 0, plot = FALSE)
smooth.df <- data.frame(freq = k1.smooth$freq, `c(4,4,4)` = k1.smooth$spec)
names(smooth.df) <- c("frequency", "c(3,3,3)")
# Add other smooths
k2 <- kernel("daniell", c(3,3))
smooth.df[, "c(3,3)"] <- spec.pgram(newdata, kernel = k2, log = 'no', taper = 0, plot = FALSE)$spec
k3 <- kernel("daniell", c(3))
smooth.df[, "c(3)"] <- spec.pgram(newdata, kernel = k3, log = 'no', taper = 0, plot = FALSE)$spec



# Melt dataframe in order to plot three graph together

smooth.df <- melt(smooth.df, variable.name = "dimension",
                  id.vars = "frequency", value.name = "spectrum")
plot1 <- ggplot(data = subset(smooth.df)) + geom_path(aes(x = frequency, y = spectrum, color = dimensio
plot2 <- ggplot(data = subset(smooth.df)) +
  geom_path(aes(x = frequency, y = spectrum, color = dimension)) +
  scale_x_continuous("Period (years)") +  scale_y_log10() +
```

```
  labs(title = "log spacing on frequency")

grid.arrange(plot1, plot2)
```




```
#Effect of different kernel

k1 <- kernel("daniell", 3, 6)
k1.smooth <- spec.pgram(newdata, kernel = k1, log = 'no', taper = 0, plot = FALSE)
smooth.df <- data.frame(freq = k1.smooth$freq, `daniell` = k1.smooth$spec)
names(smooth.df) <- c("frequency", "daniell")
# Add other smooths kernel
k2 <- kernel("dirichlet", 3, 6)   # Q : NaN ???
smooth.df[, "dirichlet"] <- spec.pgram(newdata, kernel = k2, log = 'no', taper = 0, plot = FALSE)$spec
```

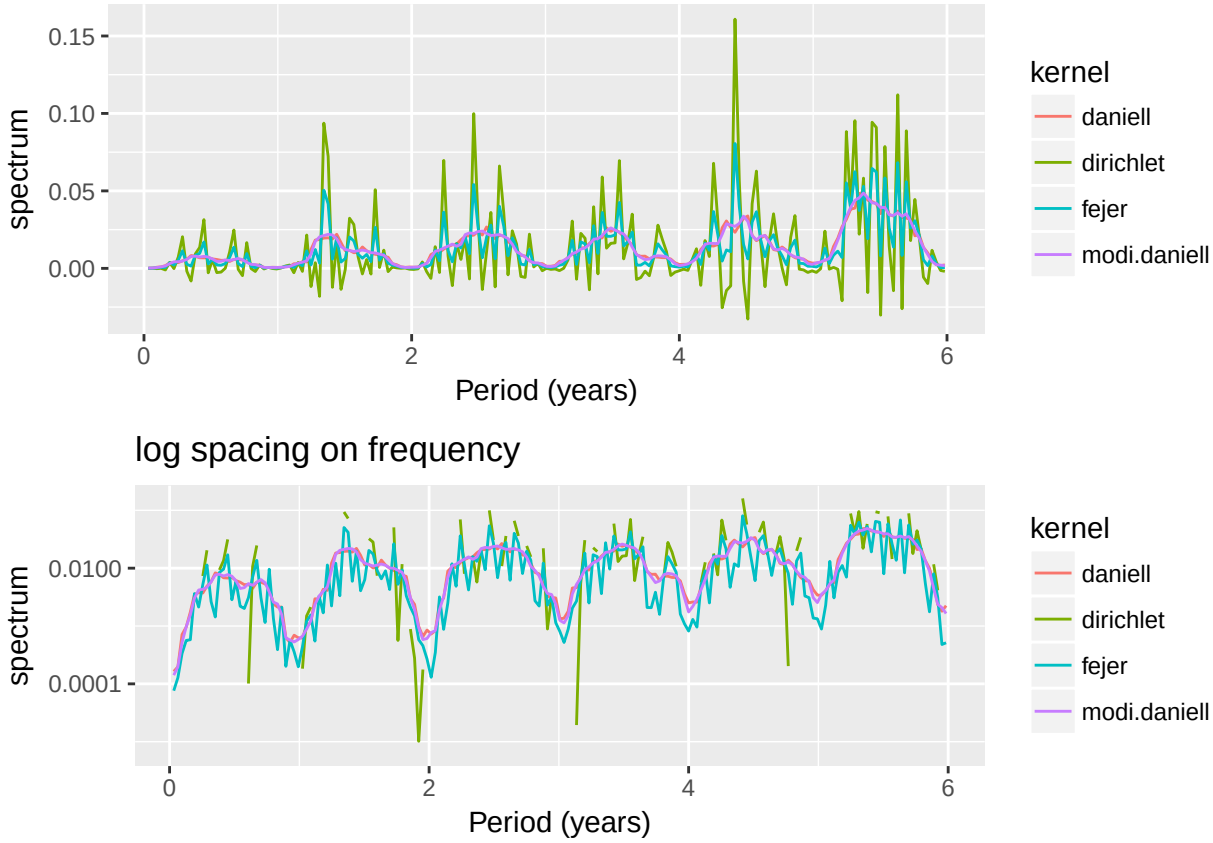
```
## Warning in sqrt(sum((1/12 + i^2) * k[i])): NaNs produced
```

```
k3 <- kernel("fejer", 3, 6)
smooth.df[, "fejer"] <- spec.pgram(newdata, kernel = k3, log = 'no', taper = 0, plot = FALSE)$spec
k4 <- kernel("modified.daniell", 3, 6)
smooth.df[, "modi.daniell"] <- spec.pgram(newdata, kernel = k4, log = 'no', taper = 0, plot = FALSE)$spe


# Melt dataframe in order to plot three graph together
smooth.df <- melt(smooth.df, variable.name = "kernel",
                  id.vars = "frequency", value.name = "spectrum")

which(is.nan(smooth.df[,3]))
```

```
## integer(0)
```

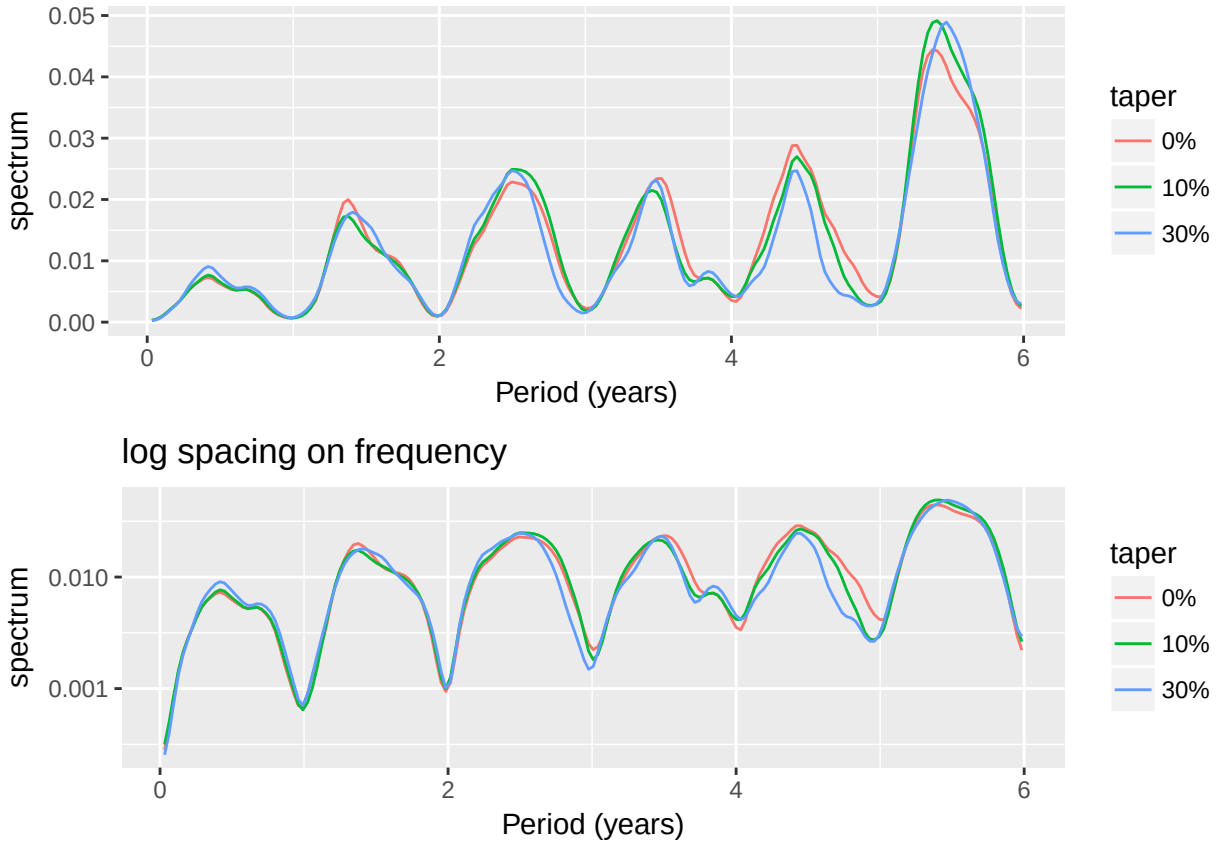
```
plot1 <- ggplot(data = subset(smooth.df)) + geom_path(aes(x = frequency, y = spectrum, color = kernel))

plot2 <- ggplot(data = subset(smooth.df)) +
  geom_path(aes(x = frequency, y = spectrum, color = kernel)) +
  scale_x_continuous("Period (years)") +  scale_y_log10() +
  labs(title = "log spacing on frequency")

grid.arrange(plot1, plot2)
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 3 rows containing missing values (geom_path).
```



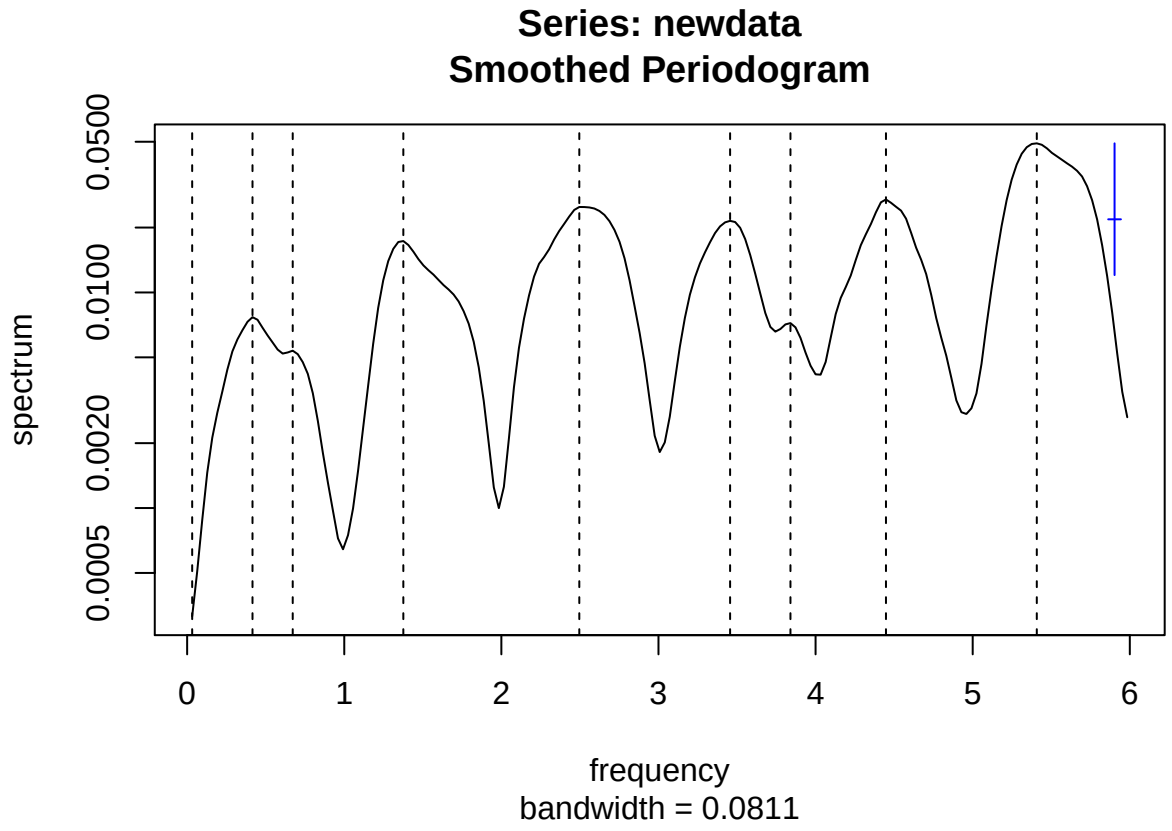

```
# Effect of different tapering - less important the longer your time series is, but it can be very impo

#Besides windowing, there is one other 'trick' commonly done when spectral estimating, called tapering.


k1 <- kernel("modified.daniell", c(3,3) , 6)

k1.smooth <- spec.pgram(newdata, kernel = k1, taper = 0, log = 'no', plot = FALSE)
smooth.df <- data.frame(freq = k1.smooth$freq, `0%` = k1.smooth$spec)
names(smooth.df) <- c("frequency", "0%")
# Add other tapers
smooth.df[, "10%"] <- spec.pgram(newdata, kernel = k1, taper = 0.1, log = 'no', plot = FALSE)$spec
```

```
smooth.df[, "30%"] <- spec.pgram(newdata, kernel = k1, taper = 0.3, log = 'no',  plot = FALSE)$spec

smooth.df <- melt(smooth.df, variable.name = "taper",
                  id.vars = "frequency", value.name = "spectrum")
plot1 <- ggplot(data = subset(smooth.df)) + geom_path(aes(x = frequency, y = spectrum, color = taper)) +
# +  scale_y_log10()

plot2 <- ggplot(data = subset(smooth.df)) + geom_path(aes(x = frequency, y = spectrum, color = taper)) +

grid.arrange(plot1, plot2)
```

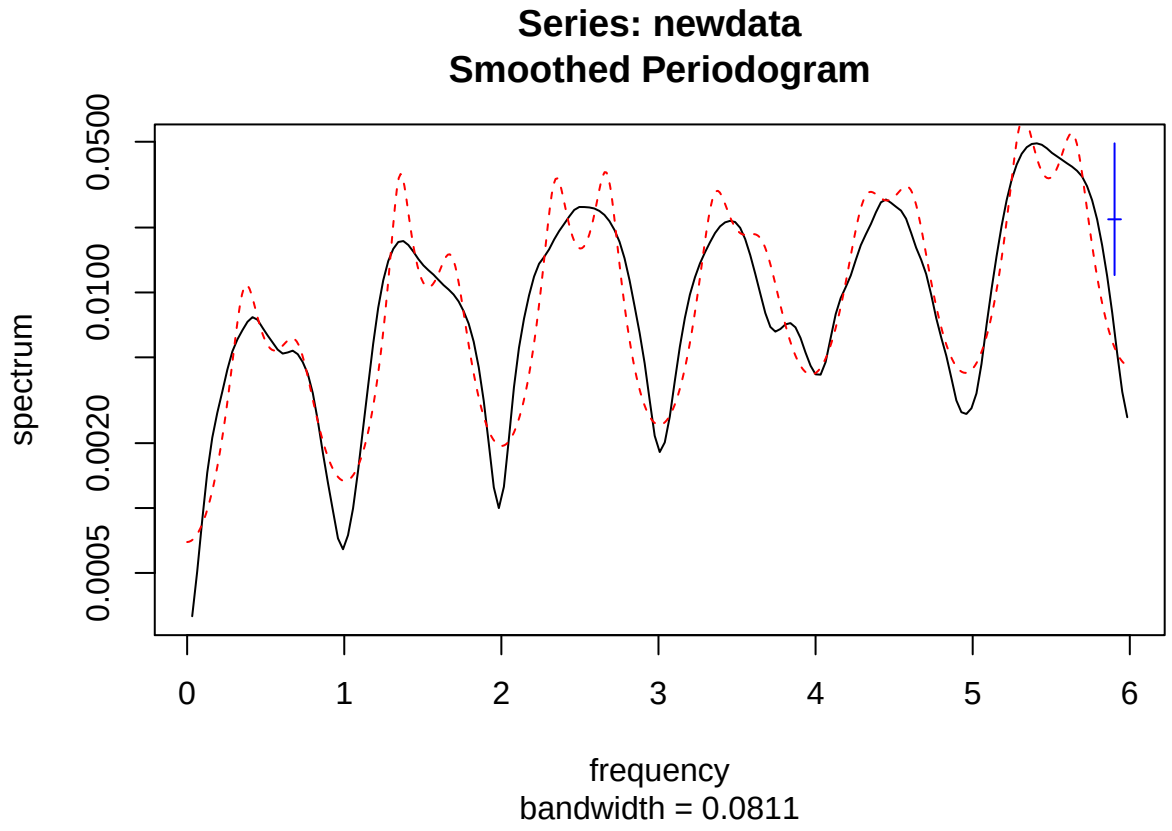


Pick the key frequencies & confidence interval

```
#pick the key frequencies

final <- spec.pgram(newdata, kernel("modified.daniell", c(3, 3)),
                    taper=0.1)
key_freq_ind <- c(1, which(diff(sign(diff(final$spec)))==-2) + 1)
key_freq <- final$freq[key_freq_ind]
abline(v=key_freq, lty=2)
```



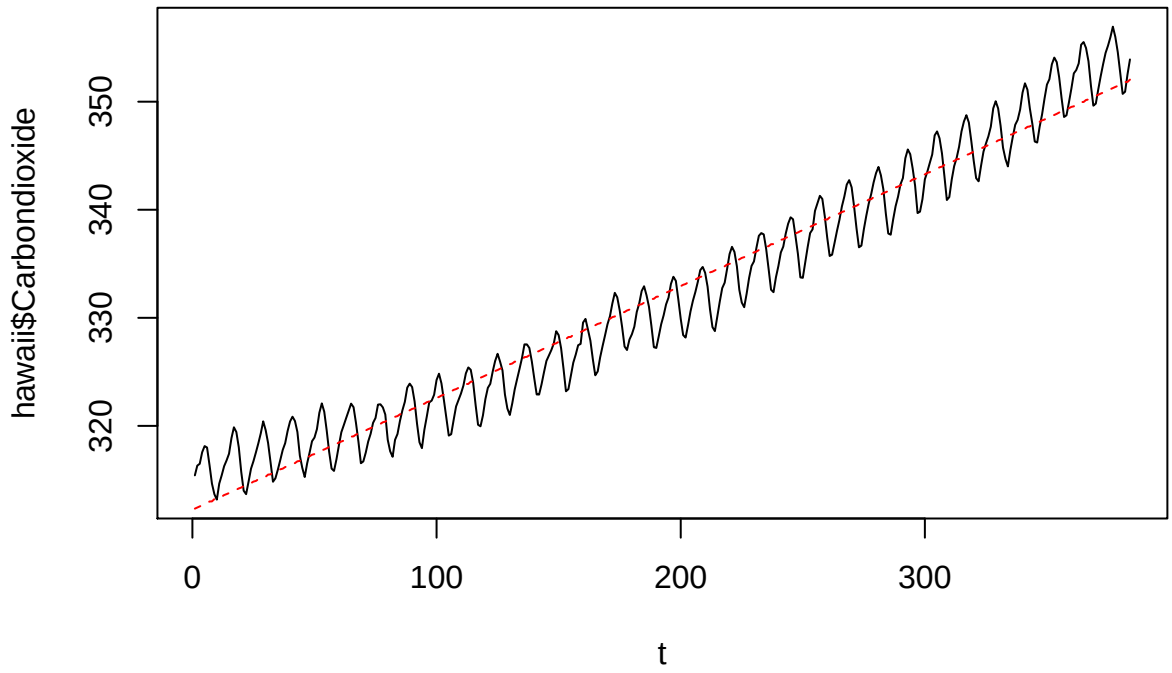

```
# compare with the parametric spectral estimator
plot(final)
final_ar <- spec.ar(newdata, plot=F)
lines(final_ar$freq, final_ar$spec, lty=2, col="red")
```




```
# pick top three frequencies and use these to generate features in terms of sin and cos functions.

t <- 1:length(hawaii[,2])

top_freq <- key_freq[order(final$spec[key_freq_ind], decreasing = T)][1:3]


top_freq
```

```
## [1] 5.408 4.448 2.496
```

```
periodic_terms <- do.call(cbind, lapply(top_freq, function(freq) {
  cbind(cos(2 * pi * freq * t), sin(2 * pi * freq * t))
}))
df <- data.frame(hawaii$Carbondioxide, t, periodic_terms)
fit_final <- lm(hawaii.Carbondioxide ~ ., df)
plot(t, hawaii$Carbondioxide, type="l")
lines(t, fit_final$fitted.values, lty=2, col="red")
```




```
k1 <- kernel("daniell", 2, 6)


final.smooth <- spec.pgram(newdata, kernel = k1, taper = 0.1, log = 'no', plot = FALSE)

final.freq <- final.smooth$freq[which.max(final.smooth$spec)]


final.freq
```

```
## [1] 5.312
```

```
df <- final.smooth$df
U <- df / qchisq(0.025, df)
L <- df / qchisq(0.975, df)
len <- length(final.smooth$spec)
idx <- round(len/5)   # Q spectral density at 0.1?

# Return Confidence Interval

c(final.smooth$spec[idx] * L, final.smooth$spec[idx] * U)
```

```
## [1] 0.001412918 0.010121668
```

Comment:

In here, we are going to use all the data (without splitting into training and testing), since we are not going to perform forecasting in spectral analysis.