STAT 4550 Project

Omar Ebrahim, Kareem El Touny, Ahmed Khaled

2024-05-06

Importing the libraries

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.3.3
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

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

```
## Registered S3 methods overwritten by 'forecast':
##   method                 from
##   autoplot.Arima         ggfortify
##   autoplot.acf           ggfortify
##   autoplot.ar            ggfortify
##   autoplot.bats          ggfortify
##   autoplot.decomposed.ts ggfortify
##   autoplot.ets           ggfortify
##   autoplot.forecast      ggfortify
##   autoplot.stl           ggfortify
##   autoplot.ts            ggfortify
##   fitted.ar              ggfortify
##   fortify.ts             ggfortify
##   residuals.ar           ggfortify
```

```
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 4.3.3
```

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

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

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(readr)
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.3.3
```

```r
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.3.2
```

UKDriverDeaths is a time series giving the monthly totals of car drivers in Great Britain killed or seriously injured Jan 1969 to Dec 1984. Compulsory wearing of seat belts was introduced on 31 Jan 1983. We chose this dataset because it is variable by time, and there seems to be a trend which allows us to analyze it in the upcoming data visualization techniques.

```r
data(Seatbelts)
head(Seatbelts)
```

```
##          DriversKilled drivers front rear   kms PetrolPrice VanKilled law
## Jan 1969           107    1687   867  269  9059   0.1029718        12   0
## Feb 1969            97    1508   825  265  7685   0.1023630         6   0
## Mar 1969           102    1507   806  319  9963   0.1020625        12   0
## Apr 1969            87    1385   814  407 10955   0.1008733         8   0
## May 1969           119    1632   991  454 11823   0.1010197        10   0
## Jun 1969           106    1511   945  427 12391   0.1005812        13   0
```

```r
Seatbelts <- as.data.frame(Seatbelts)
Seatbelts
```

```
##    DriversKilled drivers front rear   kms PetrolPrice VanKilled law
## 1            107    1687   867  269  9059  0.10297181        12   0
## 2             97    1508   825  265  7685  0.10236300         6   0
## 3            102    1507   806  319  9963  0.10206249        12   0
## 4             87    1385   814  407 10955  0.10087330         8   0
## 5            119    1632   991  454 11823  0.10101967        10   0
## 6            106    1511   945  427 12391  0.10058119        13   0
## 7            110    1559  1004  522 13460  0.10377398        11   0
## 8            106    1630  1091  536 14055  0.10407640         6   0
## 9            107    1579   958  405 12106  0.10377398        10   0
## 10           134    1653   850  437 11372  0.10302640        16   0
## 11           147    2152  1109  434  9834  0.10273011        13   0
## 12           180    2148  1113  437  9267  0.10199719        14   0
## 13           125    1752   925  316  9130  0.10127456        14   0
## 14           134    1765   903  311  8933  0.10070398         6   0
## 15           110    1717  1006  351 11000  0.10013961         8   0
## 16           102    1558   892  362 10733  0.09862110        11   0
## 17           103    1575   990  486 12912  0.09834929         7   0
## 18           111    1520   866  429 12926  0.09808018        13   0
## 19           120    1805  1095  551 13990  0.09727921        13   0
## 20           129    1800  1204  646 14926  0.09741062        11   0
## 21           122    1719  1029  456 12900  0.09742524        11   0
## 22           183    2008  1147  475 12034  0.09638063        14   0
## 23           169    2242  1171  456 10643  0.09573896        16   0
## 24           190    2478  1299  468 10742  0.09510631        14   0
## 25           134    2030   944  356 10266  0.09673597        17   0
## 26           108    1655   874  271 10281  0.09610922        16   0
## 27           104    1693   840  354 11527  0.09536725        15   0
## 28           117    1623   893  427 12281  0.09470959        13   0
## 29           157    1805  1007  465 13587  0.09411762        13   0
## 30           148    1746   973  440 13049  0.09353215        15   0
## 31           130    1795  1097  539 16055  0.09295405        12   0
## 32           140    1926  1194  646 15220  0.09283979         6   0
## 33           136    1619   988  457 13824  0.09272474         9   0
## 34           140    1992  1077  446 12729  0.09226965        13   0
## 35           187    2233  1045  402 11467  0.09170669        14   0
## 36           150    2192  1115  441 11351  0.09126207        15   0
## 37           159    2080  1005  359 10803  0.09071160        14   0
## 38           143    1768   857  334 10548  0.09027633         3   0
## 39           114    1835   879  312 12368  0.08995192        12   0
## 40           127    1569   887  427 13311  0.08909964        13   0
## 41           159    1976  1075  434 13885  0.08867919        12   0
## 42           156    1853  1121  486 14088  0.08815929         8   0
## 43           138    1965  1190  569 16932  0.08890206         8   0
## 44           120    1689  1058  523 16164  0.08818133        15   0
## 45           117    1778   939  418 14883  0.08894029         8   0
## 46           170    1976  1074  452 13532  0.08772661         5   0
## 47           168    2397  1089  462 12220  0.08742885        17   0
## 48           198    2654  1208  497 12025  0.08703543        14   0
## 49           144    2097   903  354 11692  0.08644992        13   0
## 50           146    1963   916  347 11081  0.08587264         5   0
## 51           109    1677   787  276 13745  0.08539822         8   0
## 52           131    1941  1114  472 14382  0.08382198         5   0
## 53           151    2003  1014  487 14391  0.08459078        12   0
## 54           140    1813  1022  505 15597  0.08413690        11   0
## 55           153    2012  1114  619 16834  0.08377841        13   0
## 56           140    1912  1132  640 17282  0.08351074        15   0
## 57           161    2084  1111  559 15779  0.08280639        11   0
## 58           168    2080  1008  453 13946  0.08117889        11   0
## 59           152    2118   916  418 12701  0.08285361        10   0
## 60           136    2150   992  419 10431  0.09419012        13   0
## 61           113    1608   731  262 11616  0.09239984         8   0
## 62           100    1503   665  299 10808  0.10816148         6   0
## 63           103    1548   724  303 12421  0.10721169         8   0
## 64           103    1382   744  401 13605  0.11404297        14   0
## 65           121    1731   910  413 14455  0.11245412        12   0
## 66           134    1798   883  426 15019  0.11131625        14   0
## 67           133    1779   900  516 15662  0.11030125        13   0
## 68           129    1887  1057  600 16745  0.10819718         9   0
## 69           144    2004  1076  459 14717  0.10702744         4   0
## 70           154    2077   919  443 13756  0.10494698        13   0
## 71           156    2092   920  412 12531  0.11935775         6   0
## 72           163    2051   953  400 12568  0.11762190        15   0
## 73           122    1577   664  278 11249  0.13302742        12   0
## 74            92    1356   607  302 11096  0.13084524        16   0
## 75           117    1652   777  381 12637  0.12831848         7   0
## 76            95    1382   633  279 13018  0.12354745        12   0
## 77            96    1519   791  442 15005  0.11858681        10   0
```

```
## 78              108    1421   790   409 15235  0.11633748         9   0
## 79              108    1442   803   416 15552  0.11516148         9   0
## 80              106    1543   884   511 16905  0.11450120         6   0
## 81              140    1656   769   393 14776  0.11352298         7   0
## 82              114    1561   732   345 14104  0.11193018        13   0
## 83              158    1905   859   391 12854  0.11061053        14   0
## 84              161    2199   994   470 12956  0.11527439        13   0
## 85              102    1473   704   266 12177  0.11379349        14   0
## 86              127    1655   684   312 11918  0.11234958        11   0
## 87              125    1407   671   300 13517  0.11175347        11   0
## 88              101    1395   643   373 14417  0.10964252        10   0
## 89               97    1530   771   412 15911  0.10844090         4   0
## 90              112    1309   644   322 15589  0.10788494         8   0
## 91              112    1526   828   458 16543  0.10908477         9   0
## 92              113    1327   748   427 17925  0.10757145        10   0
## 93              108    1627   767   346 15406  0.10616402        10   0
## 94              128    1748   825   421 14601  0.10630000         5   0
## 95              154    1958   810   344 13107  0.10482531        13   0
## 96              162    2274   986   370 12268  0.10345175        12   0
## 97              112    1648   714   291 11972  0.10144992        10   0
## 98               79    1401   567   224 12028  0.10040232         9   0
## 99               82    1411   616   266 14033  0.09886203         7   0
## 100             127    1403   678   338 14244  0.10249615         5   0
## 101             108    1394   742   298 15287  0.10302743        10   0
## 102             110    1520   840   386 16954  0.10217891         5   0
## 103             123    1528   888   479 17361  0.09983664         6   0
## 104             103    1643   852   473 17694  0.09263669         8   0
## 105              97    1515   774   332 16222  0.09181496         6   0
## 106             140    1685   831   391 14969  0.09072430        12   0
## 107             165    2000   889   370 13624  0.09002121        15   0
## 108             183    2215  1046   431 13842  0.08933071         7   0
## 109             148    1956   889   366 12387  0.08844273        14   0
## 110             111    1462   626   250 11608  0.08835257         4   0
## 111             116    1563   808   355 15021  0.08675736        10   0
## 112             115    1459   746   304 14834  0.08499524         8   0
## 113             100    1446   754   379 16565  0.08456794         7   0
## 114             106    1622   865   440 16882  0.08443190        11   0
## 115             134    1657   980   500 18012  0.08435088         3   0
## 116             125    1638   959   511 18855  0.08360098         5   0
## 117             117    1643   856   384 17243  0.08341726        11   0
## 118             122    1683   798   366 16045  0.08274514        10   0
## 119             153    2050   942   432 14745  0.08523527        10   0
## 120             178    2262  1010   390 13726  0.08477030         7   0
## 121             114    1813   796   306 11196  0.08445892        10   0
## 122              94    1445   643   232 12105  0.08535212        11   0
## 123             128    1762   794   342 14723  0.08755921         9   0
## 124             119    1461   750   329 15582  0.09038292         7   0
## 125             111    1556   809   394 16863  0.09078329         8   0
## 126             110    1431   716   355 16758  0.10874278        13   0
## 127             114    1427   851   385 17434  0.11414223         8   0
## 128             118    1554   931   463 18359  0.11299293         5   0
## 129             115    1645   834   453 17189  0.11132071         8   0
## 130             132    1653   762   373 16909  0.10912623         7   0
## 131             153    2016   880   401 15380  0.10769846        12   0
## 132             171    2207  1077   466 15161  0.10760157        10   0
## 133             115    1665   748   306 14027  0.10377502         7   0
## 134              95    1361   593   263 14478  0.10711417         4   0
## 135              92    1506   720   323 16155  0.10737477        10   0
## 136             100    1360   646   310 16585  0.11169537         4   0
## 137              95    1453   765   424 18117  0.11063818         8   0
## 138             114    1522   820   403 17552  0.11185521         8   0
## 139             102    1460   807   406 18299  0.10974234         7   0
## 140             104    1552   885   466 19361  0.10819393        10   0
## 141             132    1548   803   381 17924  0.10625536         8   0
## 142             136    1827   860   369 17872  0.10419303        14   0
## 143             117    1737   825   378 16058  0.10193397         8   0
## 144             137    1941   911   392 15746  0.10279382         9   0
## 145             111    1474   704   284 15226  0.10476034         8   0
## 146             106    1458   691   316 14932  0.10400254         6   0
## 147              98    1542   688   321 16846  0.11665552         7   0
## 148              84    1404   714   358 16854  0.11516148         6   0
## 149              94    1522   814   378 18146  0.11298954         5   0
## 150             105    1385   736   382 17559  0.11386064         4   0
## 151             123    1641   876   433 18655  0.11911808         5   0
## 152             109    1510   829   506 19453  0.12448999        10   0
## 153             130    1681   818   428 17923  0.12322295         7   0
## 154             153    1938   942   479 17915  0.12067793        10   0
## 155             134    1868   782   370 16496  0.12104898        12   0
## 156              99    1726   823   349 13544  0.11696857         7   0
```

```
## 157           115    1456   595  238 13601  0.11275026         4   0
## 158           104    1445   673  285 15667  0.10807931         5   0
## 159           131    1456   660  324 17358  0.10883852         6   0
## 160           108    1365   676  346 18112  0.11129177         4   0
## 161           103    1487   755  410 18581  0.11130401         4   0
## 162           115    1558   815  411 18759  0.11545436         8   0
## 163           122    1488   867  496 20668  0.11476830         8   0
## 164           122    1684   933  534 21040  0.11720743         3   0
## 165           125    1594   798  396 18993  0.11907640         7   0
## 166           137    1850   950  470 18668  0.11796586        12   0
## 167           138    1998   825  385 16768  0.11744913         2   0
## 168           152    2079   911  411 16551  0.11698846         7   0
## 169           120    1494   619  281 16231  0.11261054         8   0
## 170            95    1057   426  300 15511  0.11365702         3   1
## 171           100    1218   475  318 18308  0.11314445         2   1
## 172            89    1168   556  391 17793  0.11849553         6   1
## 173            82    1236   559  398 19205  0.11796940         3   1
## 174            89    1076   483  337 19162  0.11768661         7   1
## 175            60    1174   587  477 20997  0.12005924         6   1
## 176            84    1139   615  422 20705  0.11943775         8   1
## 177           113    1427   618  495 18759  0.11888127         8   1
## 178           126    1487   662  471 19240  0.11846236         4   1
## 179           122    1483   519  368 17504  0.11801660         3   1
## 180           118    1513   585  345 16591  0.11770662         5   1
## 181            92    1357   483  296 16224  0.11777609         5   1
## 182            86    1165   434  319 16670  0.11479699         3   1
## 183            81    1282   513  349 18539  0.11573525         4   1
## 184            84    1110   548  375 19759  0.11535626         3   1
## 185            87    1297   586  441 19584  0.11481536         6   1
## 186            90    1185   522  465 19976  0.11477748         6   1
## 187            79    1222   601  472 21486  0.11493598         7   1
## 188            96    1284   644  521 21626  0.11479699         5   1
## 189           122    1444   643  429 20195  0.11409316         7   1
## 190           120    1575   641  408 19928  0.11646552         7   1
## 191           137    1737   711  490 18564  0.11602611         4   1
## 192           154    1763   721  491 18149  0.11606673         7   1
```

```
Seatbelts <- ts(Seatbelts, start = 1969, frequency=12)
SeatbeltsTS <- Seatbelts[,1]
frequency(SeatbeltsTS)
```

```
## [1] 12
```

```
summary(SeatbeltsTS)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    60.0   104.8   118.5   122.8   138.0   198.0
```

```
mean(SeatbeltsTS)
```

```
## [1] 122.8021
```

```
sd(SeatbeltsTS)
```

```
## [1] 25.37989
```

```
min(SeatbeltsTS)
```

```
## [1] 60
```

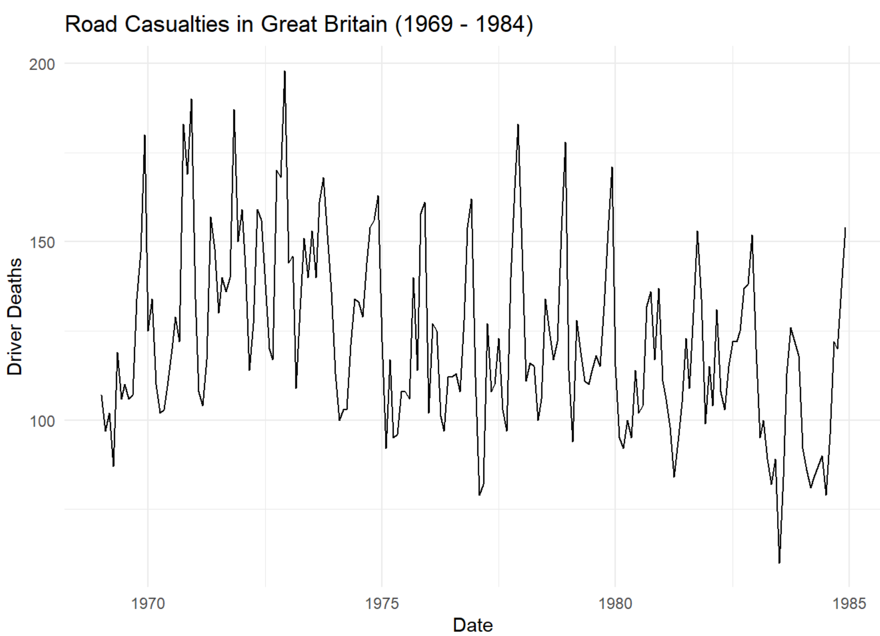
```
max(SeatbeltsTS)
```

```
## [1] 198
```

```
SeatbeltsTS
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1969 107  97 102  87 119 106 110 106 107 134 147 180
## 1970 125 134 110 102 103 111 120 129 122 183 169 190
## 1971 134 108 104 117 157 148 130 140 136 140 187 150
## 1972 159 143 114 127 159 156 138 120 117 170 168 198
## 1973 144 146 109 131 151 140 153 140 161 168 152 136
## 1974 113 100 103 103 121 134 133 129 144 154 156 163
## 1975 122  92 117  95  96 108 108 106 140 114 158 161
## 1976 102 127 125 101  97 112 112 113 108 128 154 162
## 1977 112  79  82 127 108 110 123 103  97 140 165 183
## 1978 148 111 116 115 100 106 134 125 117 122 153 178
## 1979 114  94 128 119 111 110 114 118 115 132 153 171
## 1980 115  95  92 100  95 114 102 104 132 136 117 137
## 1981 111 106  98  84  94 105 123 109 130 153 134  99
## 1982 115 104 131 108 103 115 122 122 125 137 138 152
## 1983 120  95 100  89  82  89  60  84 113 126 122 118
## 1984  92  86  81  84  87  90  79  96 122 120 137 154
```

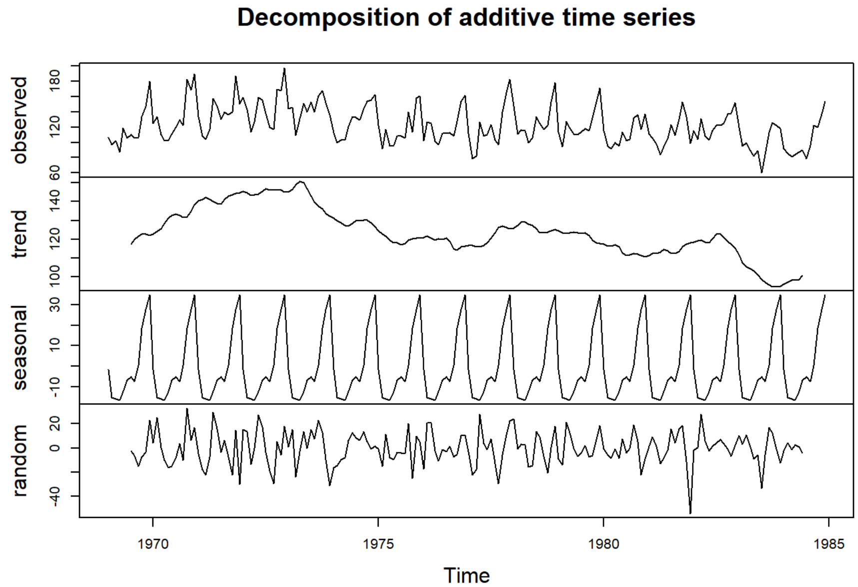
```
autoplot(SeatbeltsTS) + labs(x= "Date", y= "Driver Deaths", title="Road Casualties in Great Britain (1969 - 1984)") + theme_
minimal()
```



Over the period from 1969 to 1984, there is a noticeable downward trend in the number of driver deaths. While the overall trend is downward, there

||—————————|

```
decomposedres <- decompose(SeatbeltsTS)
plot(decomposedres)
```



| In the observed section, there is a downward trend over the years particularly for the mean, the fluctuations in the variation remain the same. T
|----------|

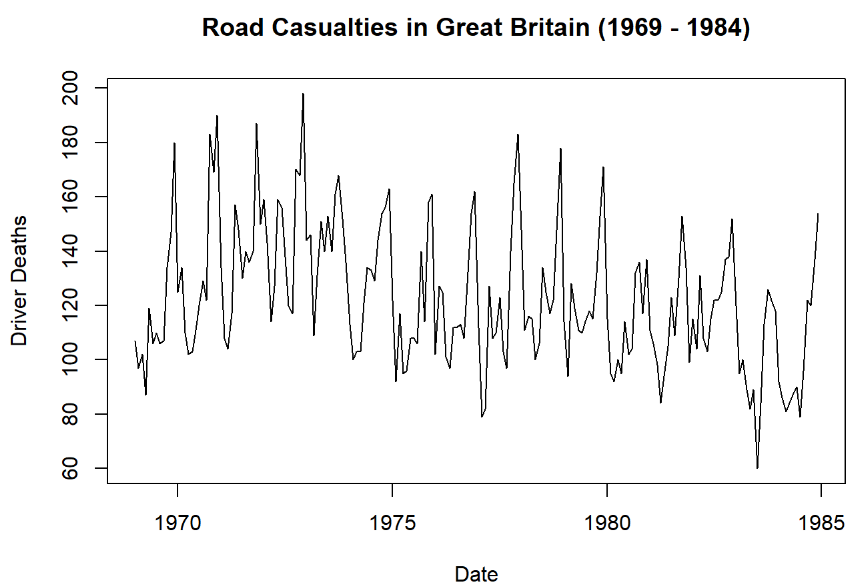
```
data(Seatbelts)
Sb <- Seatbelts[,1]
class(Sb)
```

```
## [1] "ts"
```

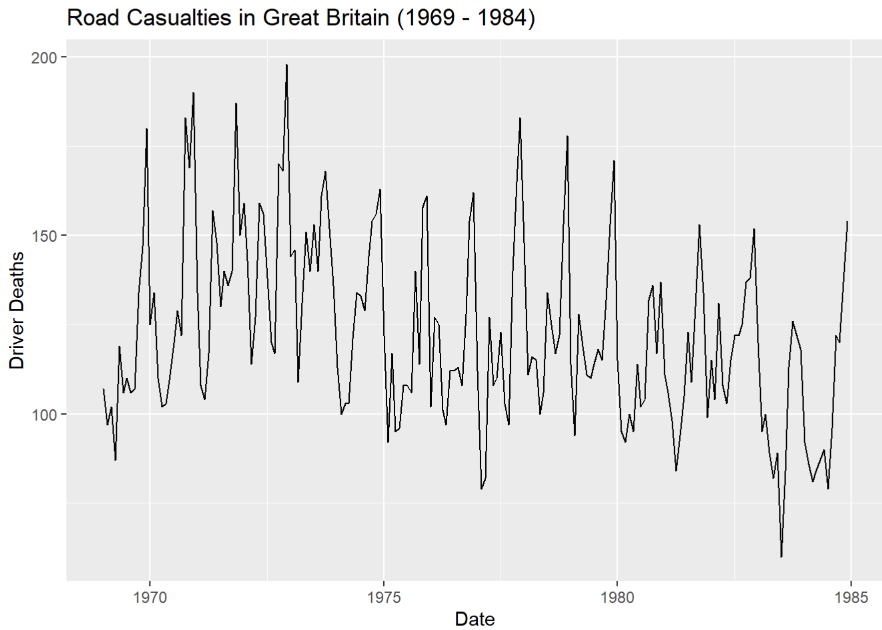
```
summary(Sb)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    60.0   104.8   118.5   122.8   138.0   198.0
```

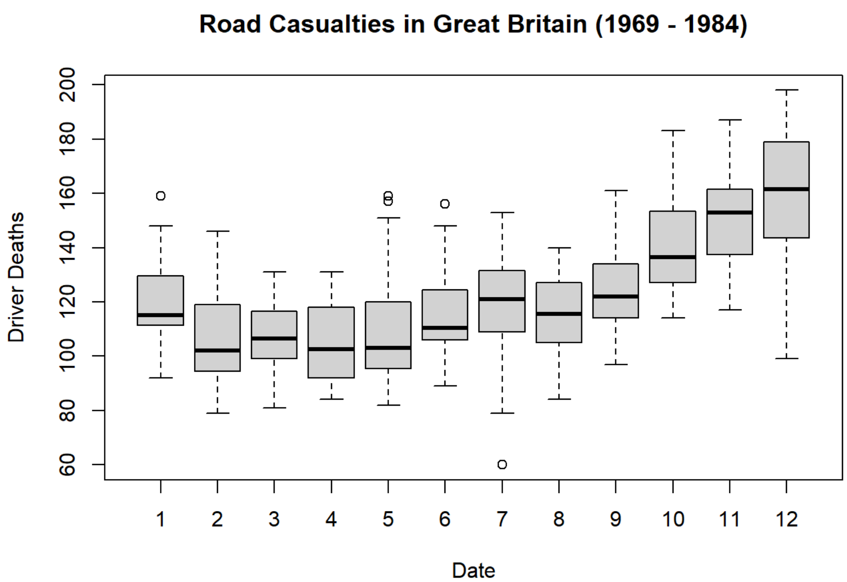
```
plot(Sb,xlab="Date", ylab = "Driver Deaths",main="Road Casualties in Great Britain (1969 - 1984)")
```



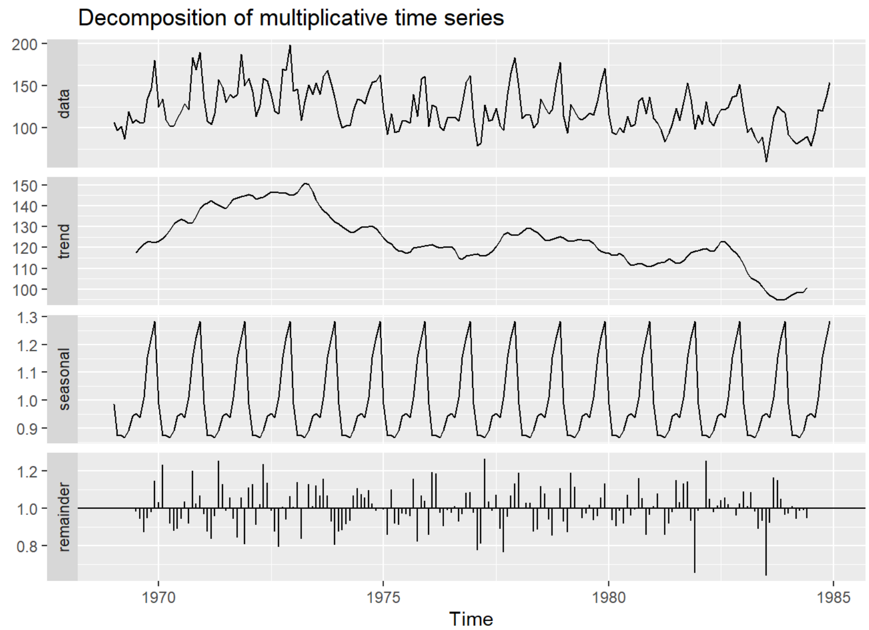

```
autoplot(Sb) + labs(x ="Date", y = "Driver Deaths", title="Road Casualties in Great Britain (1969 - 1984)")
```




```
boxplot(Sb~cycle(Sb),xlab="Date", ylab = "Driver Deaths" ,main ="Road Casualties in Great Britain (1969 - 1984)")
```




```
decomposeTS <- decompose(Sb, "multiplicative")
autoplot(decomposeTS)
```



In the data panel for the last graph, there is an upward trend, then followed by a downward trend, with clear seasonal fluctuations throughout the

| |———————————| ### Durbin-Watson (DW) Test:

```
library(lmtest)
lm_model <- lm(Sb ~ 1)
dwtest(lm_model)
```

```
##
##  Durbin-Watson test
##
## data:  lm_model
## DW = 0.73963, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

The DW statistic is from 0 to 4. A value around 2 is no autocorrelation, less than 2 positive autocorrelation, greater than 2 negative autocorrela

Dickey-Fuller (DF) Test:

```
# Load necessary package if not already loaded
library(tseries)

# Perform Dickey-Fuller test
adf.test(Sb)
```

```
## Warning in adf.test(Sb): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Sb
## Dickey-Fuller = -6.1975, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

For Dickey-Fuller, the test statistic -6.1975 is quite negative and shows strong evidence against the null hypothesis of unit root. Next, lag or

Augmented Dickey-Fuller (ADF) Test:

```
adf_result <- adf.test(Sb)
```

```
## Warning in adf.test(Sb): p-value smaller than printed p-value
```

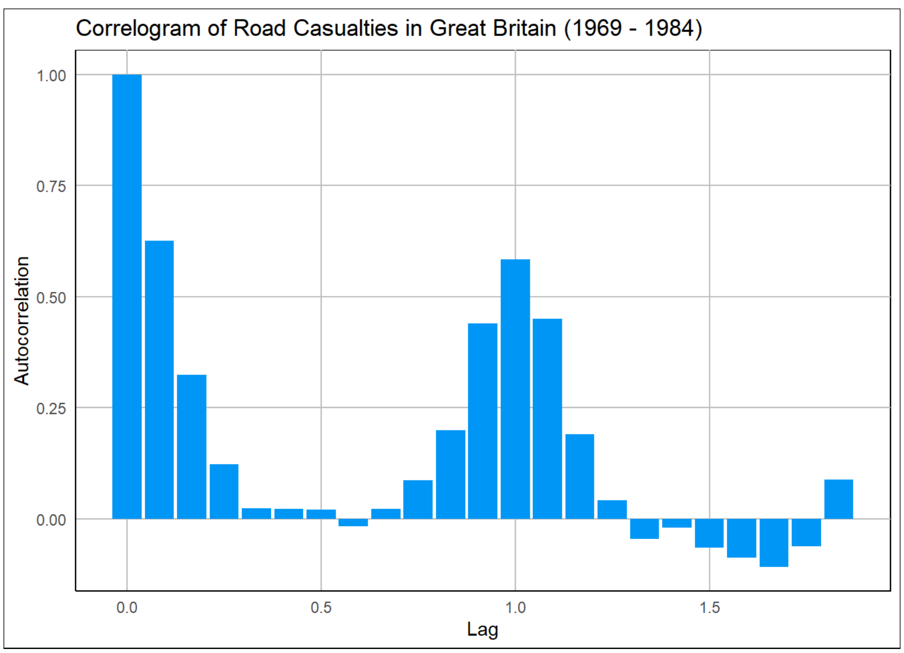
```
print(adf_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Sb
## Dickey-Fuller = -6.1975, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

The Augmented Dickey-Fuller test results show that the series is stationary, meaning that the mean & variance remain constant over time. -6.1975 :

Autocorrelation Function (ACF):

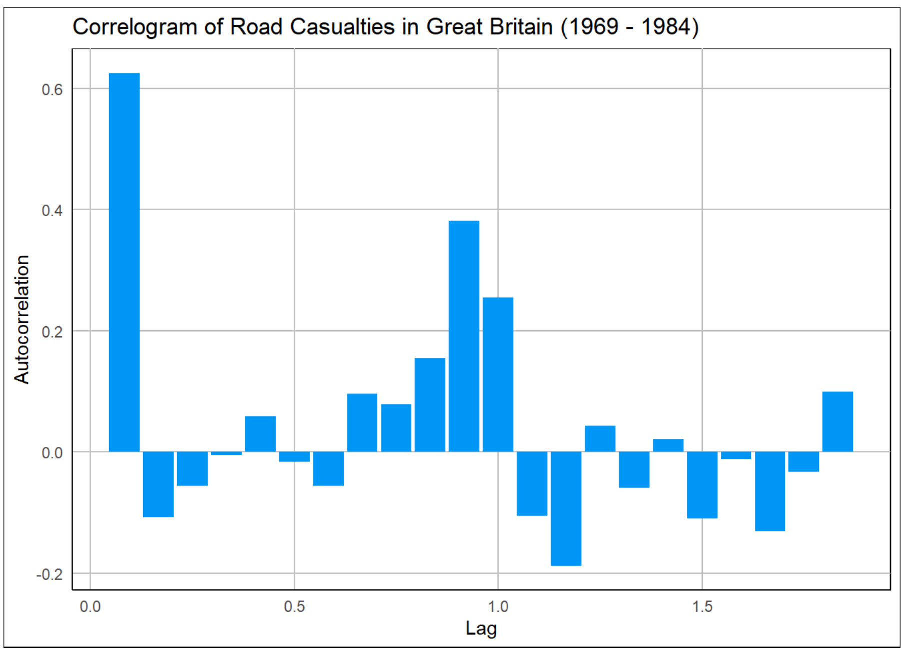
```r
library(ggplot2)  # Load the ggplot2 package for plotting
library(stats)    # Load the stats package for acf function

# Calculate autocorrelation function (ACF)
acf_result <- acf(Sb, plot = FALSE)

# Extract lag and autocorrelation values
lags <- acf_result$lag
acf_values <- acf_result$acf

# Create data frame
acf_df <- data.frame(lag = lags, acf = acf_values)

# Plot ACF using ggplot2 with white background
p5<- ggplot(acf_df, aes(x = lag, y = acf)) +
  geom_bar(stat = "identity", fill = "#0099f9") +
  labs(title = "Correlogram of Road Casualties in Great Britain (1969 - 1984)",
       x = "Lag", y = "Autocorrelation") +
  theme_minimal() +
  theme(panel.background = element_rect(fill = "white"),
        plot.background = element_rect(fill = "white"),
        panel.grid.major = element_line(color = "gray"),
        panel.grid.minor = element_blank(),
        axis.line = element_line(color = "black"),
        text = element_text(color = "black"))
print(p5)
```



Significant positive autocorrelations at lower lags suggest that the road casualties data exhibit persistence which means that high casualty peri

Partial Autocorrelation Function (PACF):

```
library(ggplot2)  # Load the ggplot2 package for plotting
library(stats)    # Load the stats package for acf function

# Calculate autocorrelation function (ACF)
pacf_result <- pacf(Sb, plot = FALSE)

# Extract lag and autocorrelation values
lags <- pacf_result$lag
pacf_values <- pacf_result$acf

# Create data frame
pacf_df <- data.frame(lag = lags, pacf = pacf_values)

# Plot ACF using ggplot2 with white background
p6<- ggplot(pacf_df, aes(x = lag, y = pacf)) +
  geom_bar(stat = "identity", fill = "#0099f9") +
  labs(title = "Correlogram of Road Casualties in Great Britain (1969 - 1984)",
       x = "Lag", y = "Autocorrelation") +
  theme_minimal() +
  theme(panel.background = element_rect(fill = "white"),
        plot.background = element_rect(fill = "white"),
        panel.grid.major = element_line(color = "gray"),
        panel.grid.minor = element_blank(),
        axis.line = element_line(color = "black"),
        text = element_text(color = "black"))
print(p6)
```




```
The plot for road shows significant positive partial autocorrelations at lag 1 which means there is a strong direct relationship between driver d
```

Modeling and Forecasting TS Data

To determine the optimal order of an ARIMA model using different information criteria such as AIC (Akaike Information Criterion), BIC (Bayesian Information Criterion), FPE (Final Prediction Error), and HQ (Hannan-Quinn Criterion)

The following script will loop through different combinations of p (AR order), d (differencing), and q (MA order) parameters, fit ARIMA models for each combination, compute AIC, BIC, FPE, and HQ criteria, and then find the optimal order based on each criterion.

```r
# Load necessary packages
library(forecast)

# Define the time series data
# Replace 'AP' with your actual time series data
ts_data <- Sb

# Define the maximum values for p, d, and q
max_p <- 3  # Maximum value for AR order
max_d <- 1  # Maximum value for differencing
max_q <- 3  # Maximum value for MA order

# Initialize matrices to store AIC, BIC, FPE, and HQ values
AIC_matrix <- matrix(NA, nrow = max_p, ncol = max_q)
BIC_matrix <- matrix(NA, nrow = max_p, ncol = max_q)
FPE_matrix <- matrix(NA, nrow = max_p, ncol = max_q)
HQ_matrix <- matrix(NA, nrow = max_p, ncol = max_q)

# Loop through different combinations of p, d, and q parameters
for (p in 1:max_p) {
  for (q in 1:max_q) {
    for (d in 0:max_d) {
      # Skip combinations that result in non-invertible models
      if (p + d + q > 0) {
        tryCatch({
          # Fit ARIMA model for the current combination of p, d, and q
          arima_model <- arima(ts_data, order = c(p, d, q))

          # Compute AIC, BIC, FPE, and HQ criteria
          AIC_matrix[p, q] <- AIC(arima_model)
          BIC_matrix[p, q] <- BIC(arima_model)
          FPE_matrix[p, q] <- logLik(arima_model) * (-2 / length(ts_data))
          HQ_matrix[p, q] <- log(length(ts_data)) * (p + q + 1) - 2 * logLik(arima_model)
        }, error = function(e) {
          next
        })
      }
    }
  }
}

# Find the optimal order based on each criterion
optimal_order_AIC <- which(AIC_matrix == min(AIC_matrix), arr.ind = TRUE)
optimal_order_BIC <- which(BIC_matrix == min(BIC_matrix), arr.ind = TRUE)
optimal_order_FPE <- which(FPE_matrix == min(FPE_matrix), arr.ind = TRUE)
optimal_order_HQ <- which(HQ_matrix == min(HQ_matrix), arr.ind = TRUE)

# Print the optimal orders
cat("Optimal Order (AIC):", optimal_order_AIC, "\n")
```

```
## Optimal Order (AIC): 2 2
```

```r
cat("Optimal Order (BIC):", optimal_order_BIC, "\n")
```

```
## Optimal Order (BIC): 2 2
```

```r
cat("Optimal Order (FPE):", optimal_order_FPE, "\n")
```

```
## Optimal Order (FPE): 2 3
```

```r
cat("Optimal Order (HQ):", optimal_order_HQ, "\n")
```

```
## Optimal Order (HQ): 2 2
```

The AIC, BIC, and HQ all suggest the same model that is ARIMA(2,0,2), which makes it a potentially robust choice for the model. However, FPE crit

Automatic **ARIMA**

ARIMA forecasting captures the autocorrelation in a series and models it directly. Autocorrelation are values that show how a series relates to itself over a time series. ARIMA models are typical for outperforming exponential smoothing methods when historical data is long and non-volatile.

```
# Load the forecast package for ARIMA modeling
library(forecast)

# Fit automatic ARIMA model
auto_arima_model <- auto.arima(Sb)

# Print model summary
print(summary(auto_arima_model))
```

```
## Series: Sb
## ARIMA(1,0,2)(0,1,1)[12] with drift
##
## Coefficients:
##          ar1      ma1      ma2     sma1    drift
##       0.9497  -0.5740  -0.2048  -0.8858  -0.1225
## s.e.  0.0498   0.0872   0.0753   0.0863   0.0900
##
## sigma^2 = 251.7:  log likelihood = -759.2
## AIC=1530.4   AICc=1530.89   BIC=1549.56
##
## Training set error measures:
##                     ME     RMSE    MAE        MPE     MAPE      MASE
## Training set 0.6768785 15.14566 11.191 -0.7505822 9.355091 0.6630612
##                     ACF1
## Training set -0.00559305
```

The ARIMA(1,0,2)(0,1,1)[12] model with drift (adds constant trend to the model) is well-fitted to the data because it captures both the seasonal a

Residuals Diagnostics

To diagnose the residuals of an ARIMA model, you can use various techniques including the Augmented Dickey-Fuller (ADF) test or inspecting the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) of the residuals. In these plots, if the autocorrelation coefficients of the residuals are significant at certain lags (outside the blue shaded region in the plot), it indicates that the residuals exhibit some patterns that are not captured by the model.

Augmented Dickey-Fuller (ADF) Test for Residuals:

```
# Perform Augmented Dickey-Fuller test for residuals
adf.test(residuals(auto_arima_model))
```

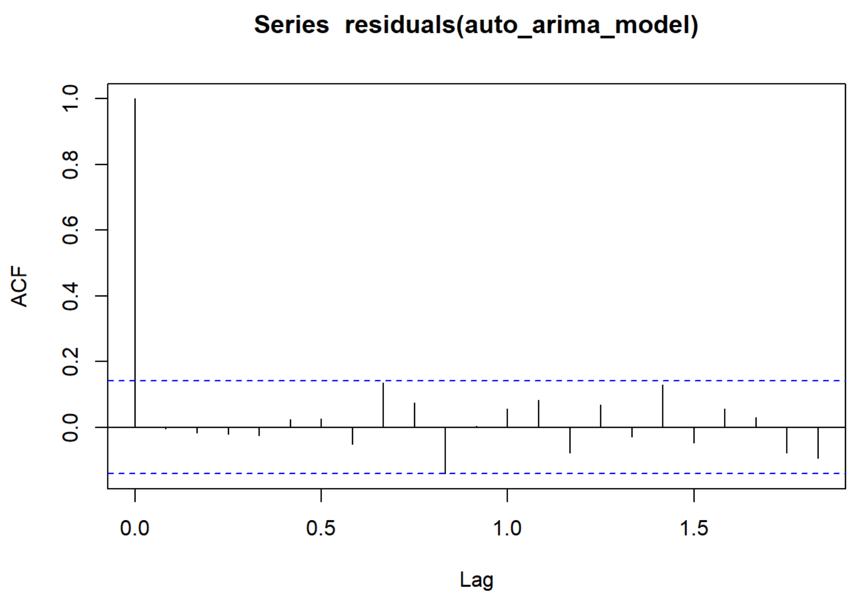
```
## Warning in adf.test(residuals(auto_arima_model)): p-value smaller than printed
## p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  residuals(auto_arima_model)
## Dickey-Fuller = -5.7588, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

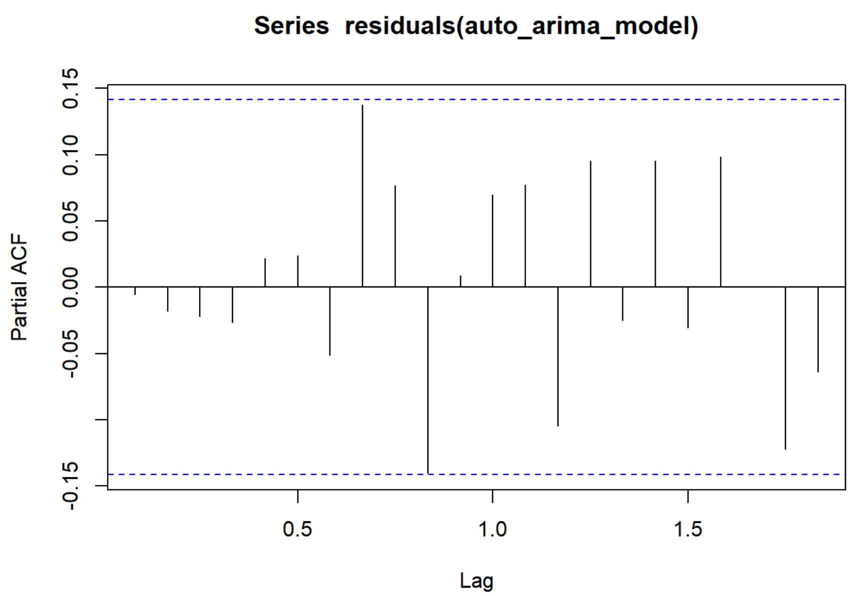
Since p-value= 0.01 < 0.05, we reject the null hypothesis that our model is non-stationary then the model is stationary.

(ACF) and (PACF) of Residuals:

```
# Plot Autocorrelation Function (ACF) of residuals
acf(residuals(auto_arima_model))
```

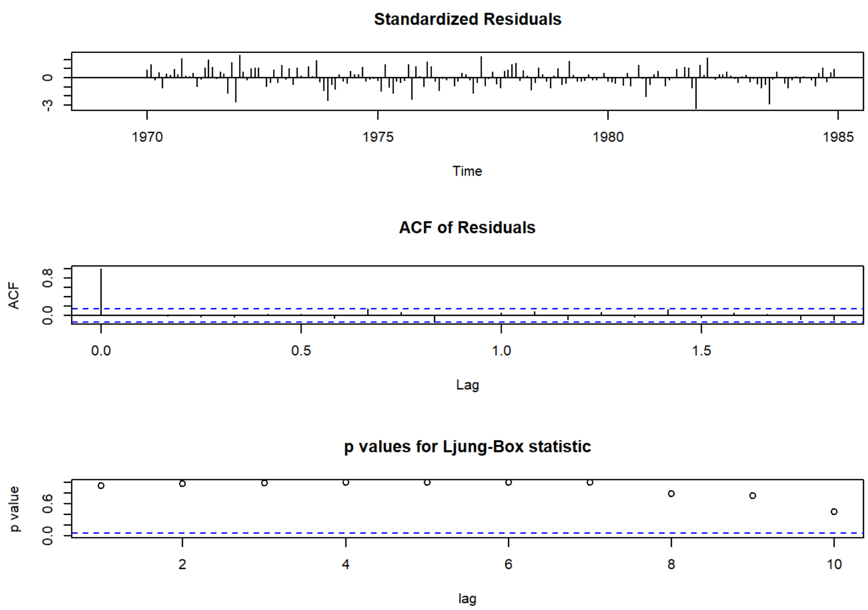



```
# Plot Partial Autocorrelation Function (PACF) of residuals
pacf(residuals(auto_arima_model))
```



Both ACF amd PACF plots show cutt-off throughout, which means there is no sign of autocorrelation in this model.

```
p6<-tsdiag(auto_arima_model)
```




```
print(p6)
```

```
## NULL
```

Standardized Residuals plot indicate having outliers which can be handled by applying some transformation or data cleaning such as checking for s

Normality

To test the normality of residuals from an ARIMA model, you can use:

1. **Shapiro-Wilk Test**:

```
# Perform Shapiro-Wilk test for normality of residuals
shapiro.test(residuals(auto_arima_model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(auto_arima_model)
## W = 0.97992, p-value = 0.007399
```

Since the p-value is less than 0.05, we reject the null hypothesis that the residuals are normally distributed. Residuals of the ARIMA model show

2. **Kolmogorov-Smirnov Test**:

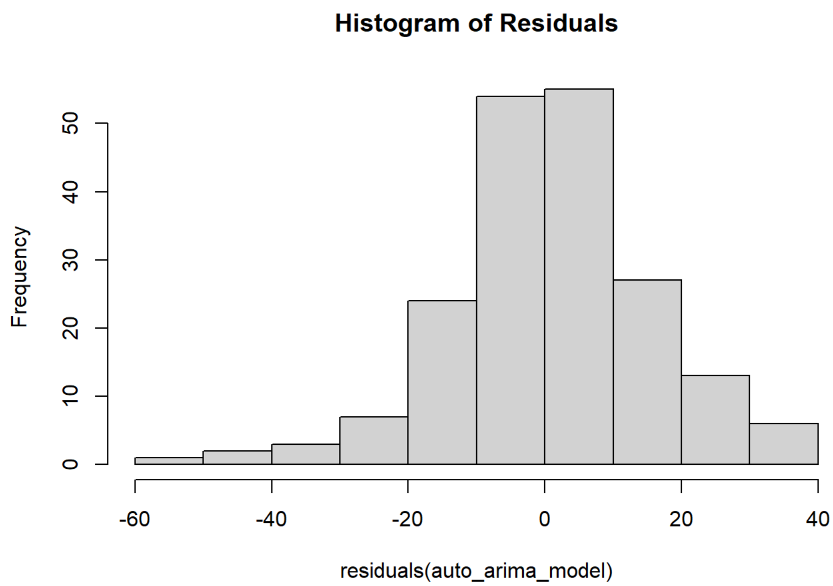
```
# Perform Kolmogorov-Smirnov test for normality of residuals
ks.test(residuals(auto_arima_model), "pnorm", mean = mean(residuals(auto_arima_model)), sd = sd(residuals(auto_arima_mode
l)))
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  residuals(auto_arima_model)
## D = 0.065639, p-value = 0.3797
## alternative hypothesis: two-sided
```

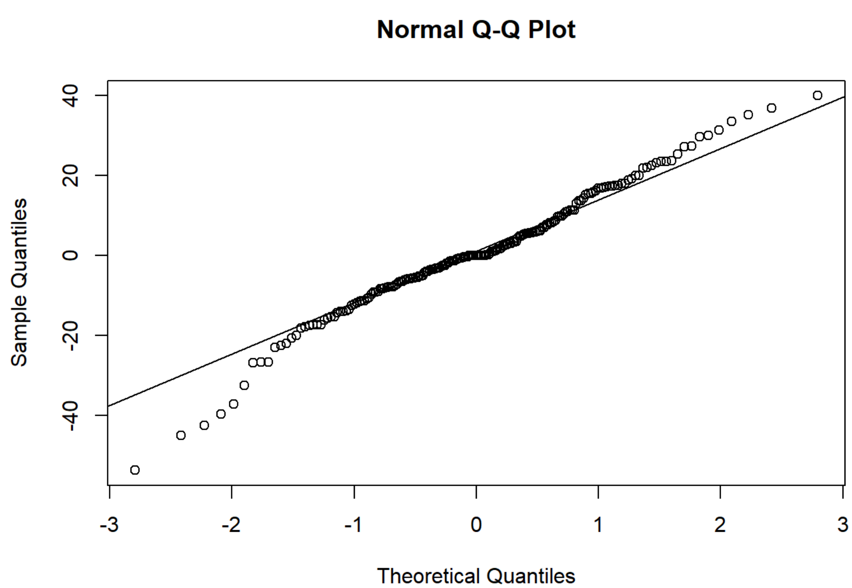
p-value is greater than 0.05 then we fail to reject H0 that states that the data follows a specific distribution which can be a normal distributi

3. **Visual Inspection**:

```
# Histogram of residuals
hist(residuals(auto_arima_model), main = "Histogram of Residuals")
```




```
# Q-Q plot of residuals
qqnorm(residuals(auto_arima_model))
qqline(residuals(auto_arima_model))
```



In the Q-Q plot, if the residuals follow a normal distribution, the points should approximately fall along the diagonal line.

By performing these tests and visual inspections, you can assess whether the residuals from your ARIMA model are approximately normally distributed.

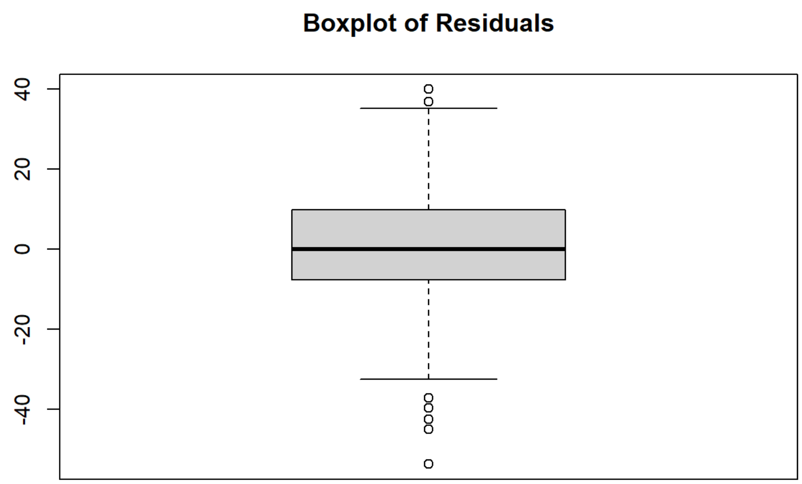
```
While both plots don't indicate a perfect normal distribution, the deviations from the 45 degree line in the Q-Q plot is not that significant to :
```

Outliers check

To check for outliers in the residuals of an ARIMA model, you can use

1. **Boxplot**:

```
# Box-plot of residuals
boxplot(resid(auto_arima_model), main = "Boxplot of Residuals")
```



`The boxplot indicates that we have 7 outliers which further verify the standardized residuals coclusion that outliers are present and more modifi`

Ljung–Box test

The Ljung-Box test is a statistical test used to check for the presence of autocorrelation in a time series at various lags.

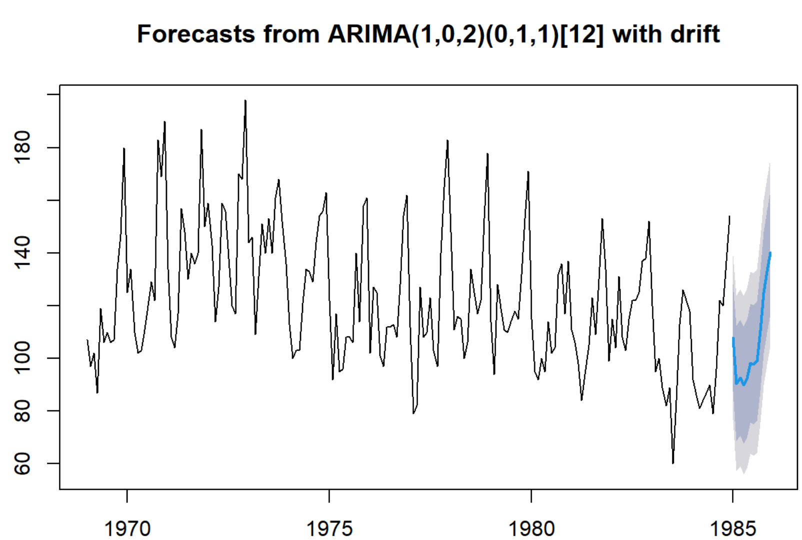
```
# Load necessary package
library(stats)

# Perform Ljung-Box test for autocorrelation of residuals
Box.test(residuals(auto_arima_model), lag = 3)
```

```
##
##  Box-Pierce test
##
## data:  residuals(auto_arima_model)
## X-squared = 0.15839, df = 3, p-value = 0.984
```

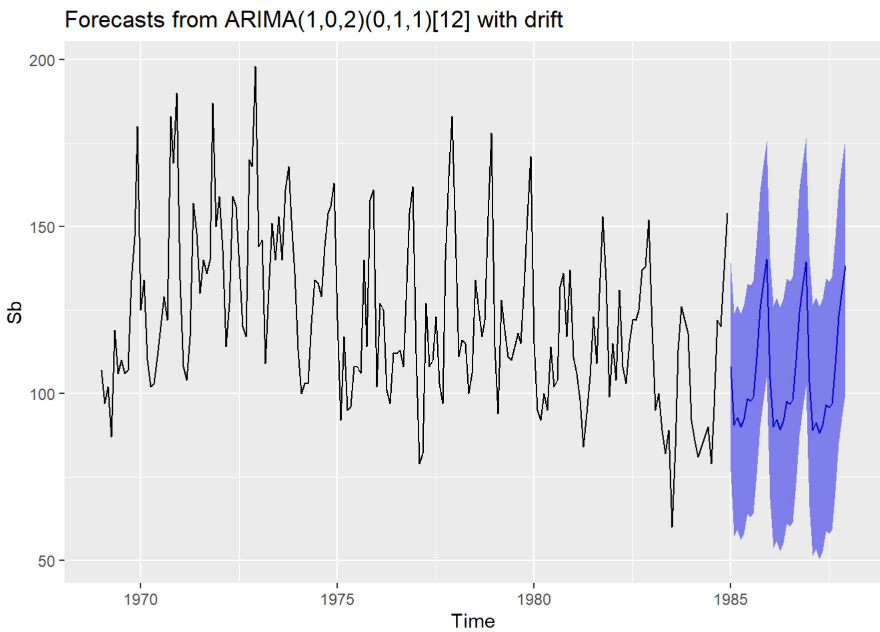
`Ljung Box test confirms that there is no autocorrelation since p-value is greater than 0.05 fail to reject H0: no autocorrelation`

ARIMA Forecasting

```
Arimaforecast <- forecast(auto_arima_model, h=12)
plot(Arimaforecast)
```



```
p6<-autoplot(Arimaforecast)

forecastSb <- forecast(auto_arima_model, level = c(95), h = 36)
p7<-autoplot(forecastSb)
print(p7)
```



The ARIMA forecast appears to capture the overall trend in the time series data, showing a gradual increase in values over time. However, there ar