

R Notebook

CASE STUDY LAB 1

```
#Set of datasets from "Forecasting: Principles and Practice" book  
library(fpp2)
```

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

```
## -- Attaching packages ----- fpp2 2.4 --
```

```
## v ggplot2  3.3.3      v fma      2.4  
## v forecast 8.13      v expsmooth 2.3
```

```
##
```

```
library(forecast)  
library(lmtest)
```

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

```
##
```

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

```
## The following objects are masked from 'package:base':
```

```
##
```

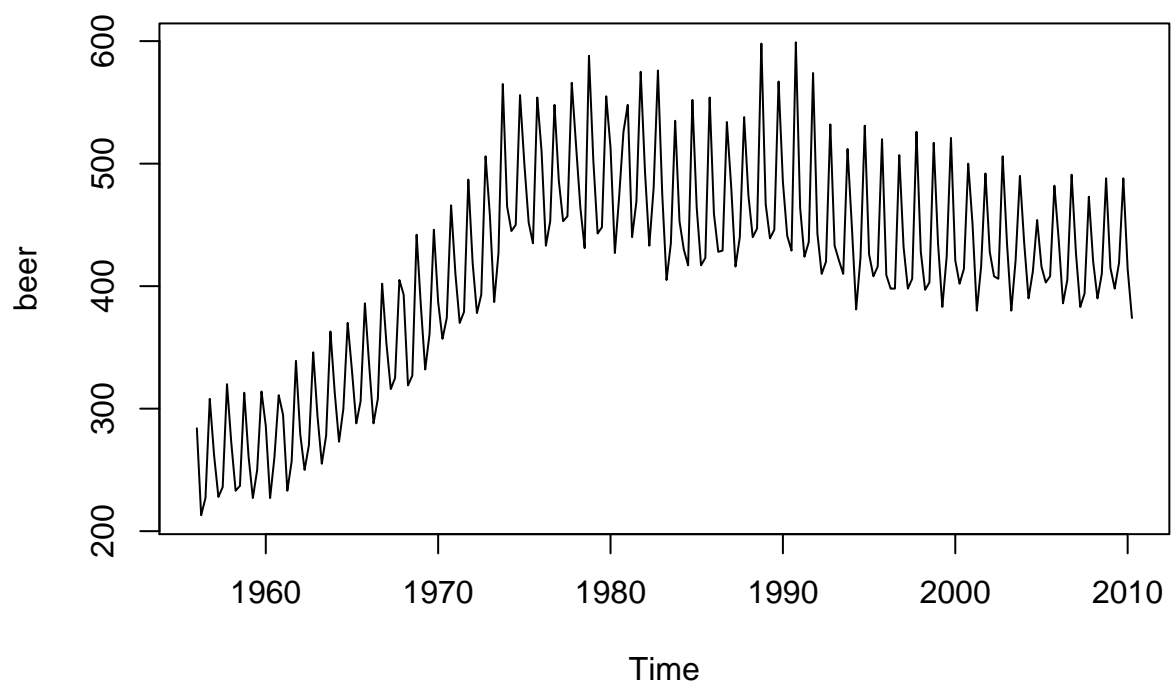
```
##   as.Date, as.Date.numeric
```

LINEAR REGRESSION with trend and seasonality

Data on Australian beer production

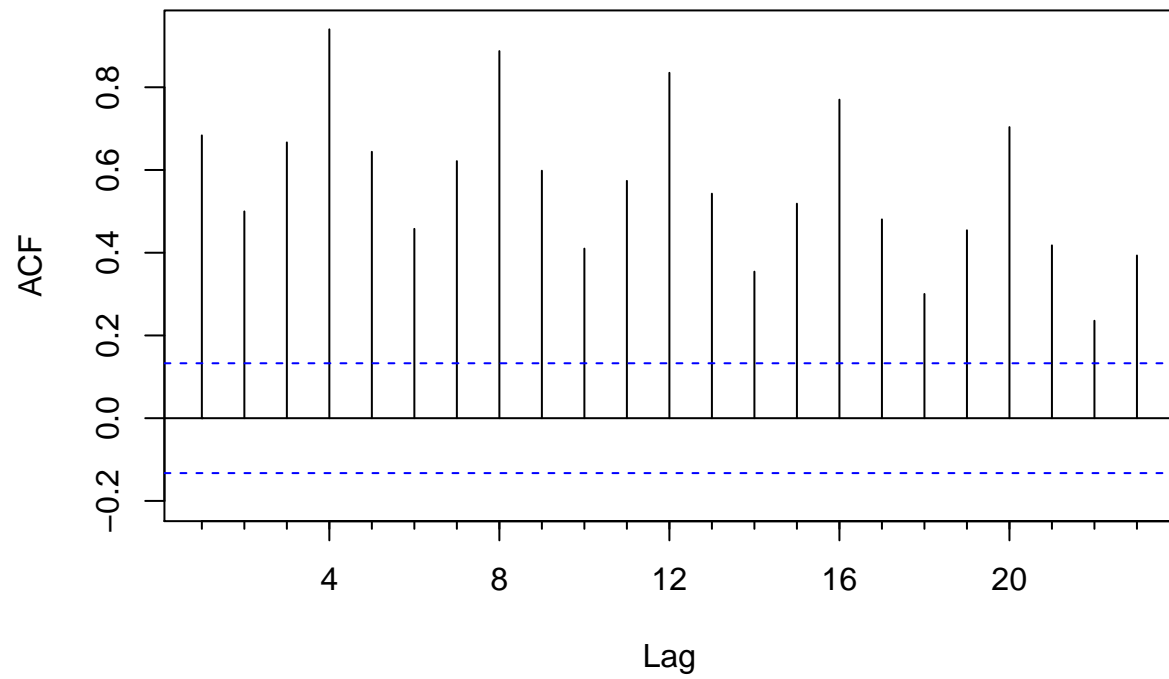
First data exploration:

```
beer<- ausbeer  
plot(beer)
```



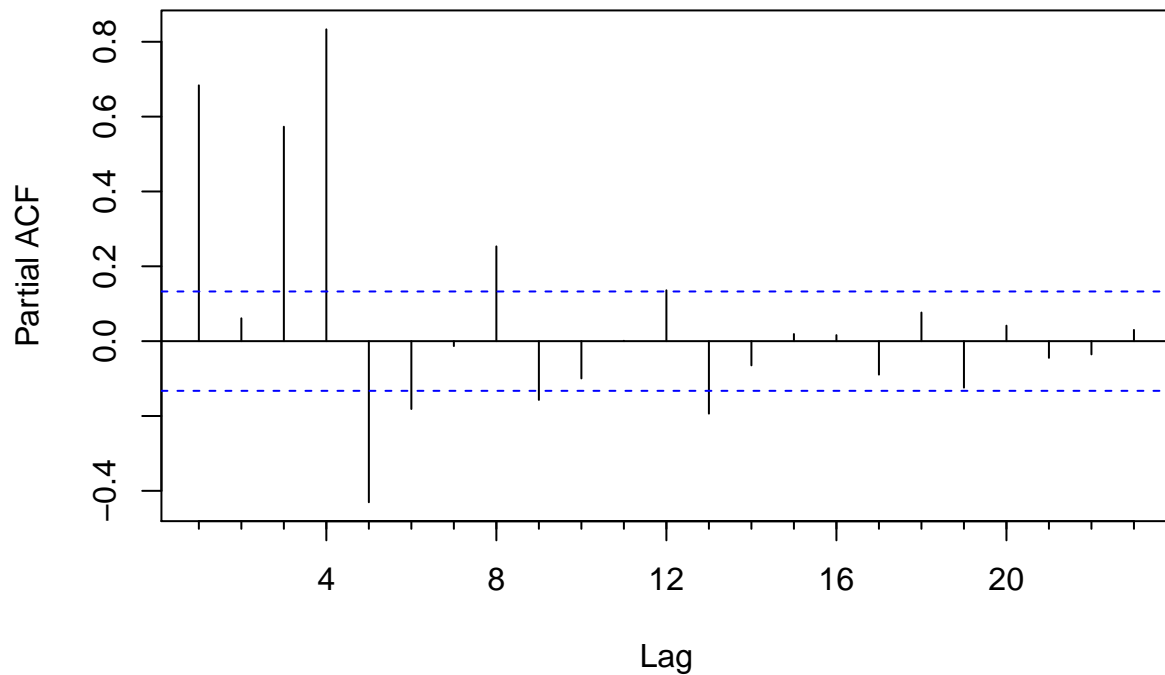
```
Acf(beer)
```

Series beer



Pacf(beer)

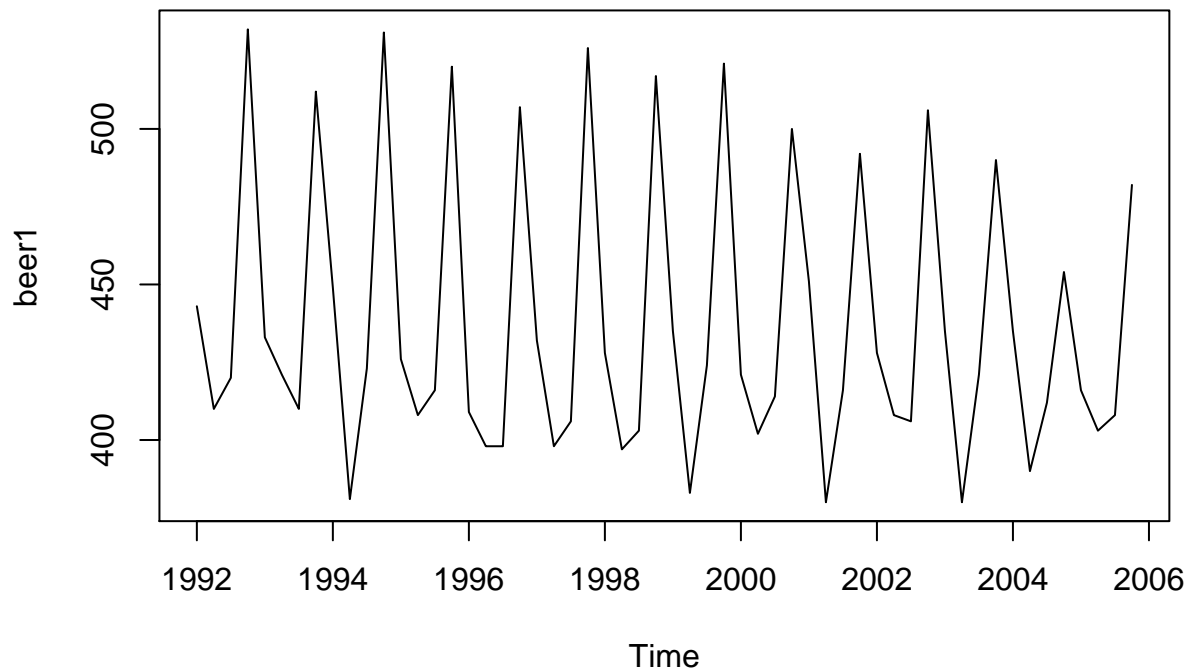
Series beer



Very strong seasonal component and trend (NON-LINEAR). Many lags appear significant, we can see a trend and a seasonality (the lags multiple of 4 are more significant) While the ACF takes into account the correlations between observations that are distant of a particular lag like, for example, the lag 4 autocorrelation is talking about the correlation between two observation that are distant 4 lags and this kind of autocorrelation takes into account also the other autocorrelation that are inside: for example autocorrelation at lag 4 takes into account also the correlation at lag 3,2 and 1, the PARTIAL autocorrelation just takes into account the pure correlation between two observation that are distant a particular lag, for example, the autocorrelation at lag 4 just takes into account the correlation between two observation that are distant 4 lags without taking into account the other correlations that are inside like 3 and 2 lags. In our PACF we can see that the lag 4 is very significant, so it confirms the seasonality

Take a portion of data and fit a linear model (to simplify the problem)

```
beer1<- window(ausbeer, start=1992, end=2006 -.1)
plot(beer1)
```



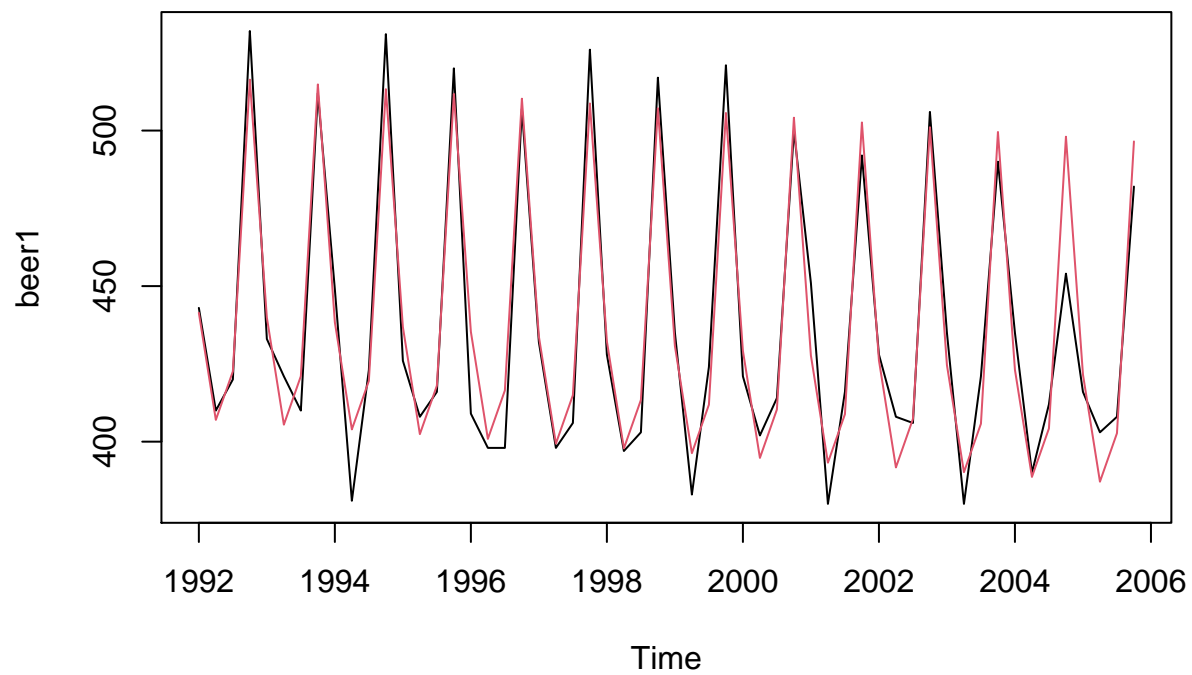
We see a strong seasonality but the NON-linear trend is removed, now we have a LINEAR trend

```
#A first linear model
m1<- tslm(beer1~ trend + season)
#We can see a slightly negative trend
summary(m1)
```

```
##
## Call:
## tslm(formula = beer1 ~ trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -44.024  -8.390   0.249   8.619  23.320
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  441.8141     4.5338  97.449 < 2e-16 ***
## trend        -0.3820     0.1078  -3.544 0.000854 ***
## season2      -34.0466     4.9174  -6.924 7.18e-09 ***
## season3      -18.0931     4.9209  -3.677 0.000568 ***
## season4       76.0746     4.9268  15.441 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.01 on 51 degrees of freedom
```

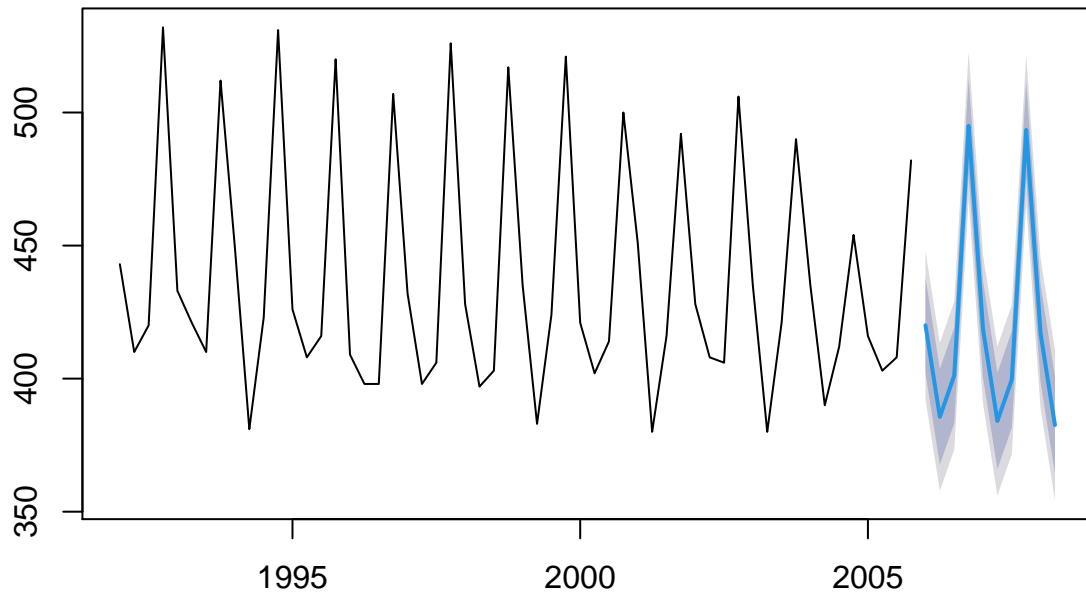
```
## Multiple R-squared:  0.921, Adjusted R-squared:  0.9149  
## F-statistic: 148.7 on 4 and 51 DF,  p-value: < 2.2e-16
```

```
fit<- fitted(m1)  
plot(beer1)  
lines(fitted(m1), col=2)
```

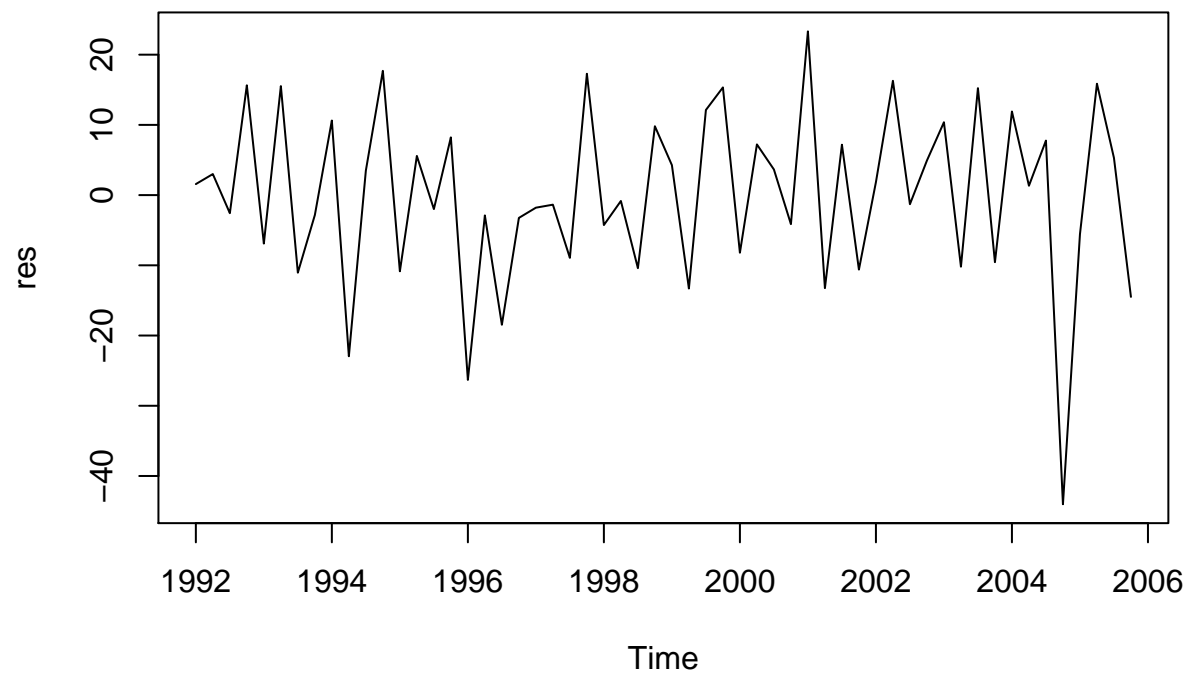


```
#Forecasting  
fore <- forecast(m1)  
plot(fore)
```

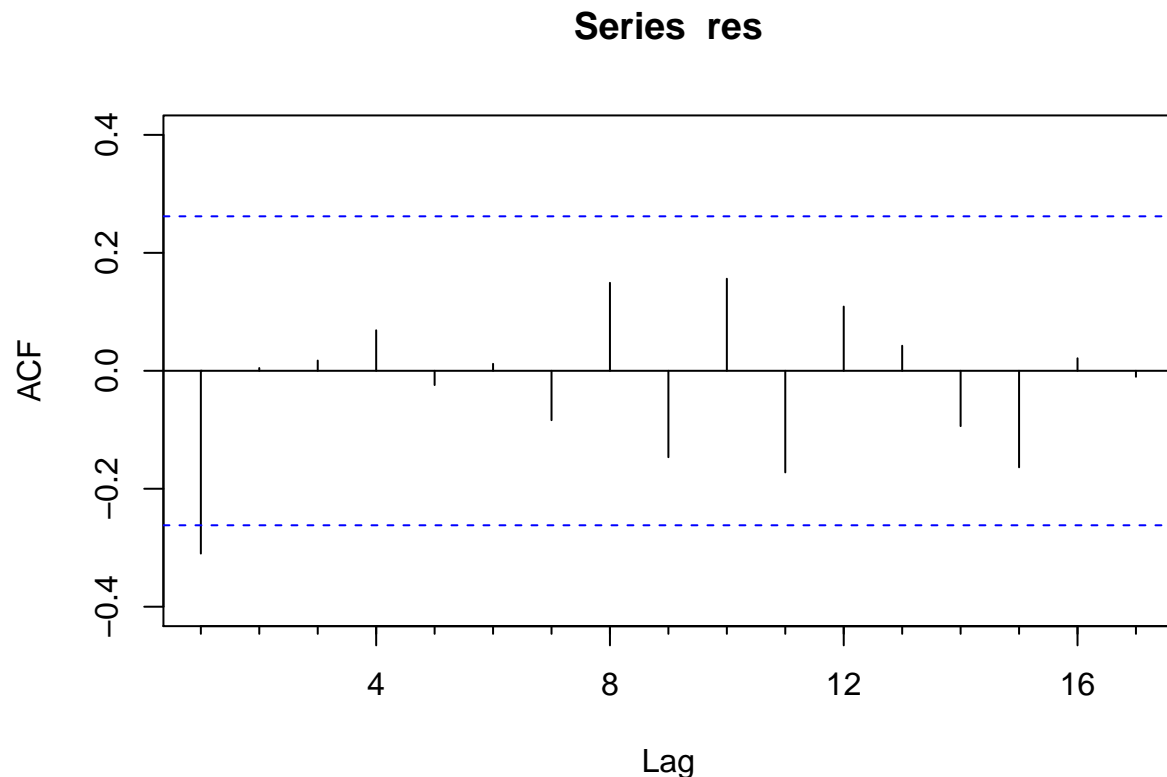
Forecasts from Linear regression model



```
#Analysis of residuals  
res<- residuals(m1)  
plot(res)
```



Acf(res)



Residuals show a particular pattern (no white noise), a seasonal behavior. Although this last fact, we can accept our model (see m1 plot) because it capture very well the behavior of our t.s.. We have an up and down behavior for every single residual. We can see a good global behavior, except for the first lag that has a NEGATIVE autocorr. This last aspect confirm the fact that if we have a first res. that is positive, the next one is negative. This is peculiar of a ZIG-ZAG behavior.

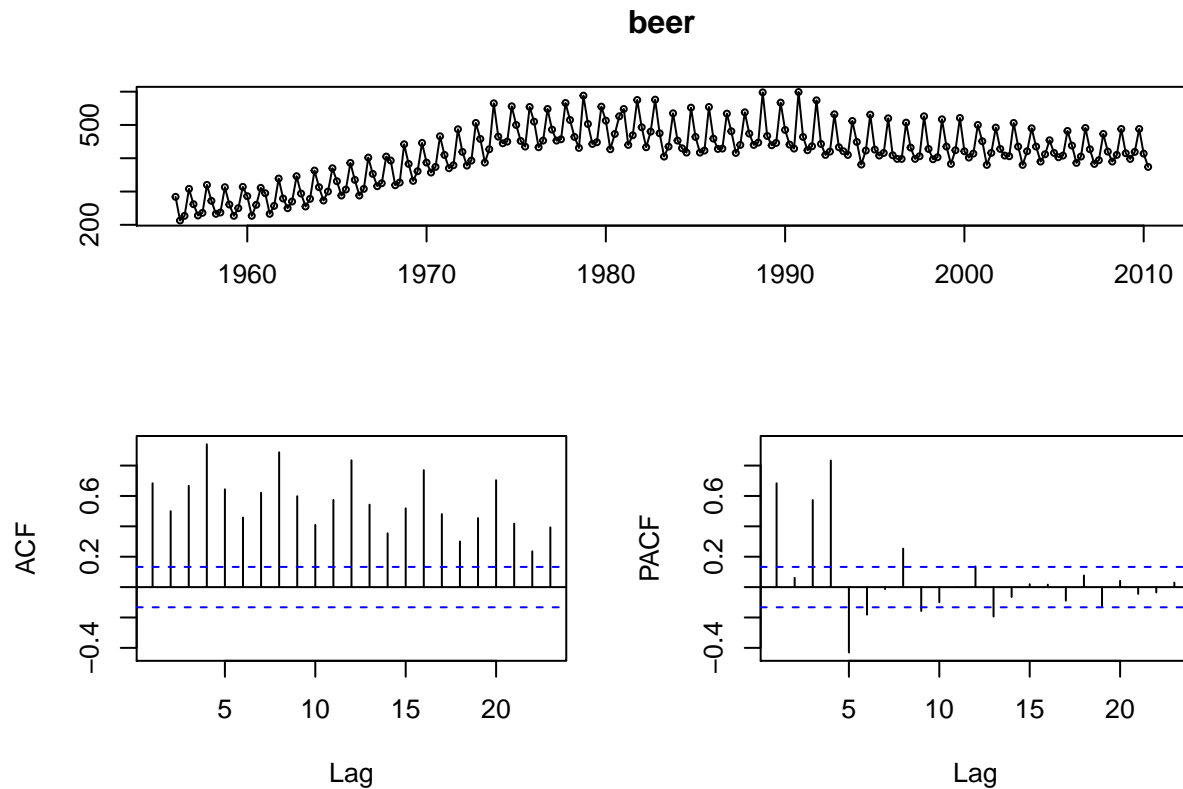
To see the significance respect the presence of autocorrelations we have the Durbin-Watson test. It detects the presence of autocorrelation in residuals of a regression model. If its value is significant we can say that the residuals are correlated each other (in a positive way if the value is between 0-2, in a negative between 2-4), otherwise they are not correlated (NB: it doesn't say which residuals are correlated, it says only a general presence of relationship). This test takes values between 0 (positive autoc.) and 4 (negative autoc.), with a central value in 2 (that indicates the absence of any autocorrelations).

```
dw<- dwtest(m1, alt="two.sided")
dw
```

```
##
## Durbin-Watson test
##
## data: m1
## DW = 2.5951, p-value = 0.02764
## alternative hypothesis: true autocorrelation is not 0
```

Exercise 1: how could we model the entire series? (To update!!! See lecture or correction of prof)

```
#Data overview
tsdisplay(beer)
```



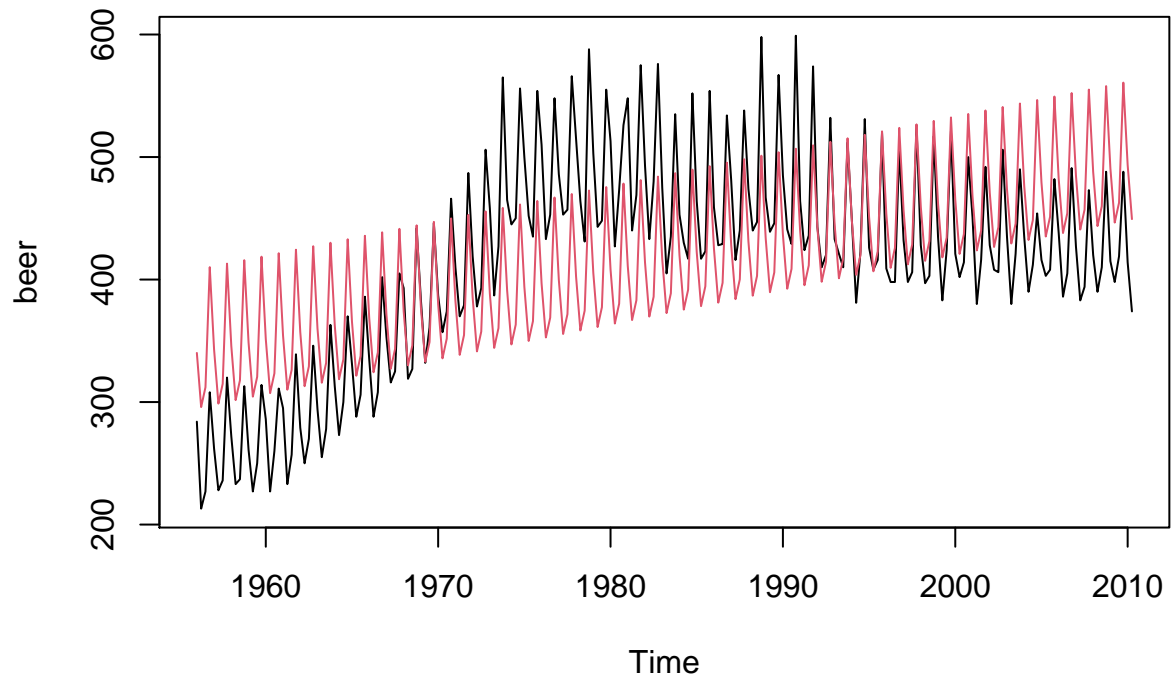
```
#Linear model
m.beer<- tslm(beer~ trend + season)
summary(m.beer)
```

```
##
## Call:
## tslm(formula = beer ~ trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -110.51  -48.99  -10.62   46.70  136.87
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 339.38485   10.65278  31.859 < 2e-16 ***
## trend        0.71040    0.06415  11.074 < 2e-16 ***
## season2     -44.91040   11.36595  -3.951 0.000106 ***
## season3     -29.44781   11.41827  -2.579 0.010582 *
```

```
## season4      67.91587   11.41845    5.948  1.1e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 59.6 on 213 degrees of freedom
## Multiple R-squared:  0.5272, Adjusted R-squared:  0.5183
## F-statistic: 59.38 on 4 and 213 DF,  p-value: < 2.2e-16
```

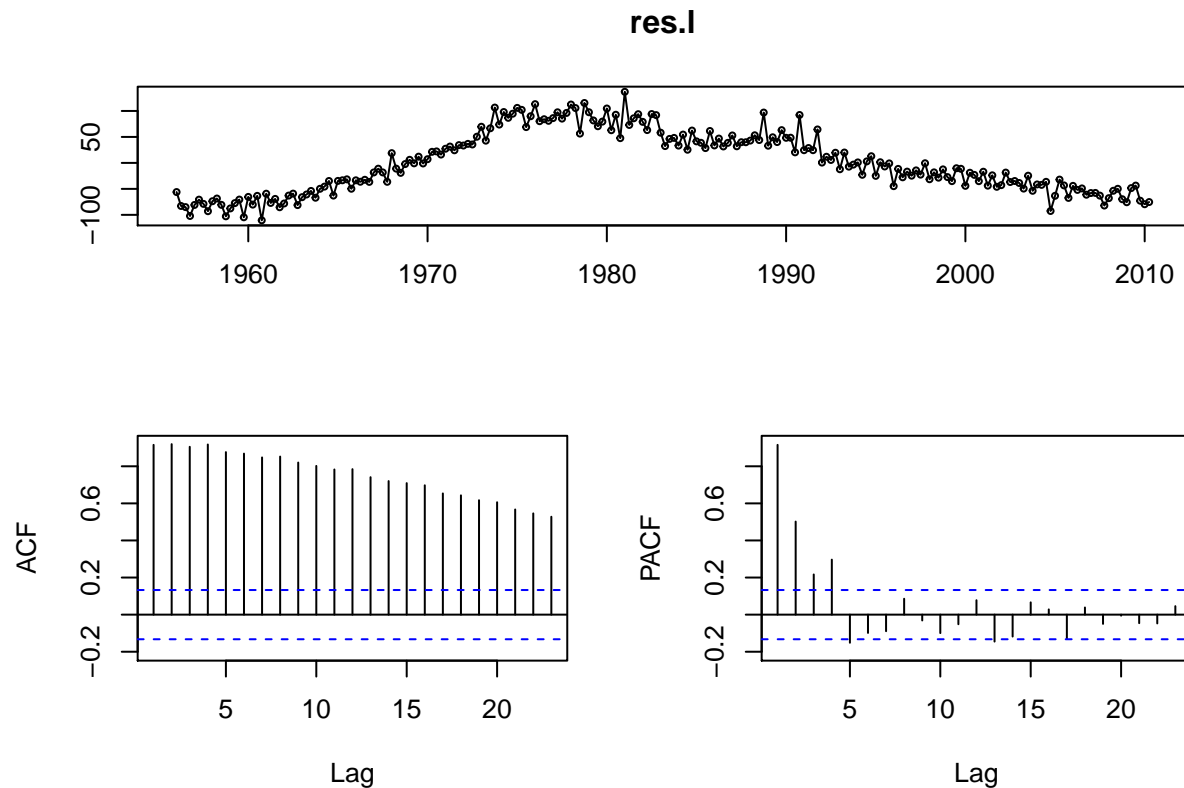
#Fitting

```
fit.l<- fitted(m.beer)
plot(beer)
lines(fit.l, col=2)
```



#Analysis of residuals

```
res.l<- residuals(m.beer)
tsdisplay(res.l)
```



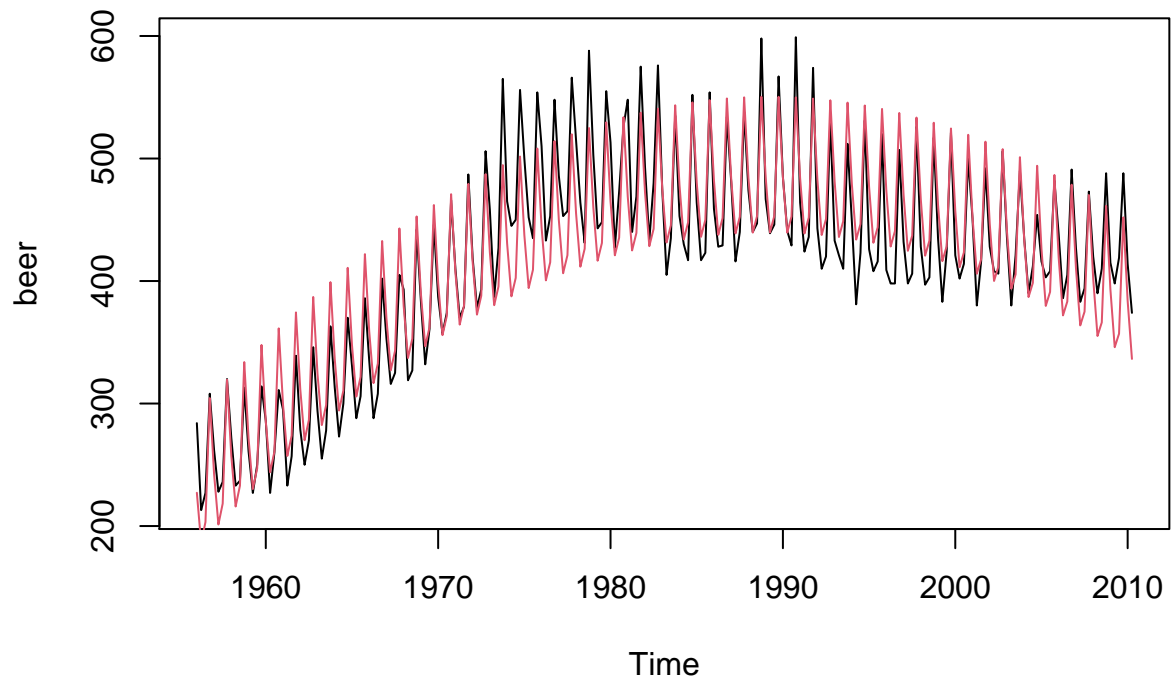
The model is too simple, it doesn't capture many informations. We can try with a polynomial regression: the data have a curved trend

```
#Linear model (polynomial version) + ARIMA on residuals
m.beer.pol<- tslm(beer~ poly(trend, 2, raw=TRUE) + season)
summary(m.beer.pol)
```

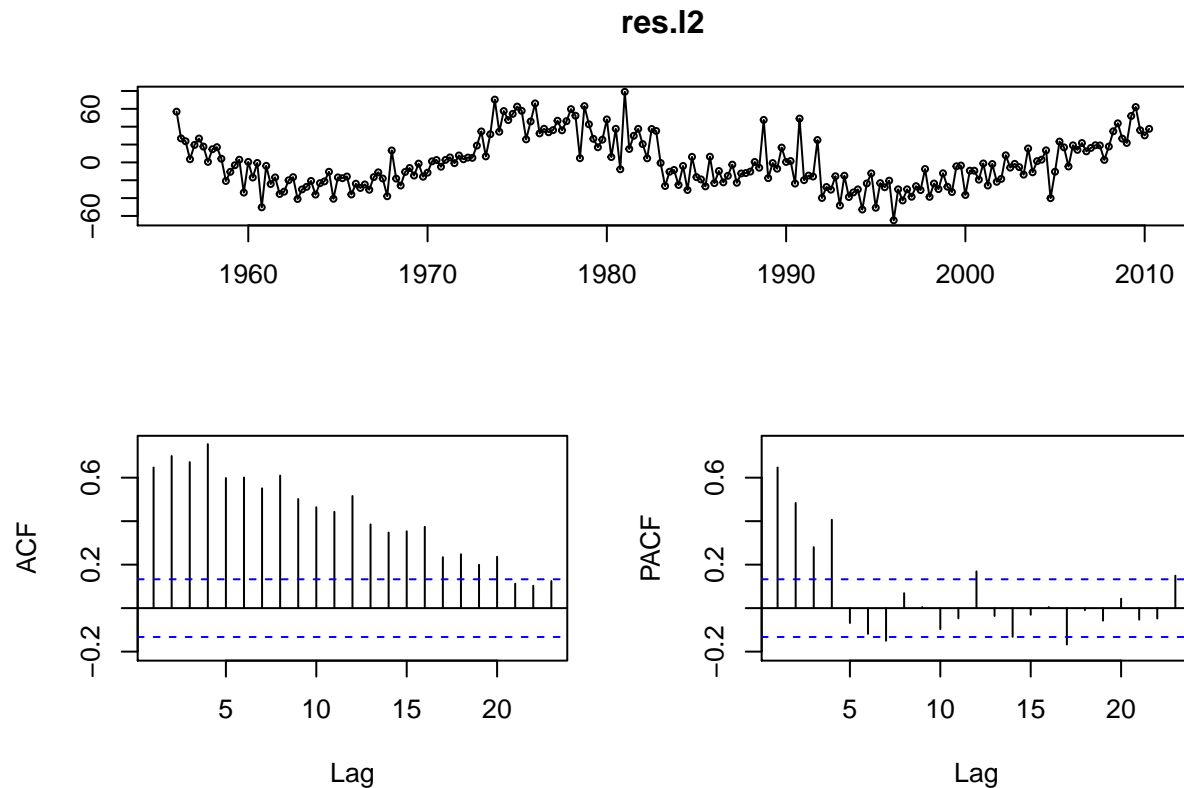
```
##
## Call:
## tslm(formula = beer ~ poly(trend, 2, raw = TRUE) + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -64.894 -21.176  -4.185  18.953  79.085
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.233e+02  6.738e+00  33.137  < 2e-16 ***
## poly(trend, 2, raw = TRUE)1  3.906e+00  1.241e-01  31.468  < 2e-16 ***
## poly(trend, 2, raw = TRUE)2 -1.459e-02  5.489e-04 -26.581  < 2e-16 ***
## season2        -4.491e+01  5.473e+00  -8.205  2.21e-14 ***
## season3        -3.157e+01  5.499e+00  -5.741  3.24e-08 ***
## season4         6.580e+01  5.499e+00  11.965  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 28.7 on 212 degrees of freedom
## Multiple R-squared:  0.8909, Adjusted R-squared:  0.8883
## F-statistic: 346.2 on 5 and 212 DF,  p-value: < 2.2e-16
```

```
#Fitting
fit.l2<- fitted(m.beer.pol)
plot(beer)
lines(fit.l2, col=2)
```



```
#Analysis of residuals
res.l2<- residuals(m.beer.pol)
tsdisplay(res.l2)
```



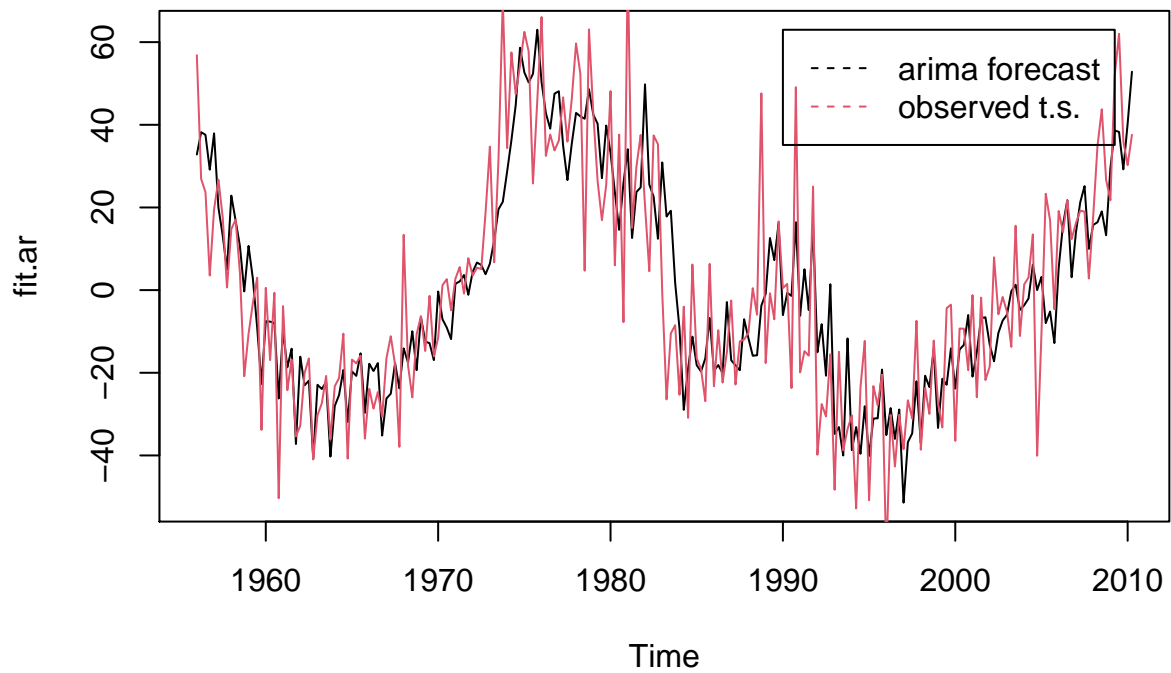
```
#ARIMA on residuals
```

```
arimaResiduals <- auto.arima(res.l2)
summary(arimaResiduals)
```

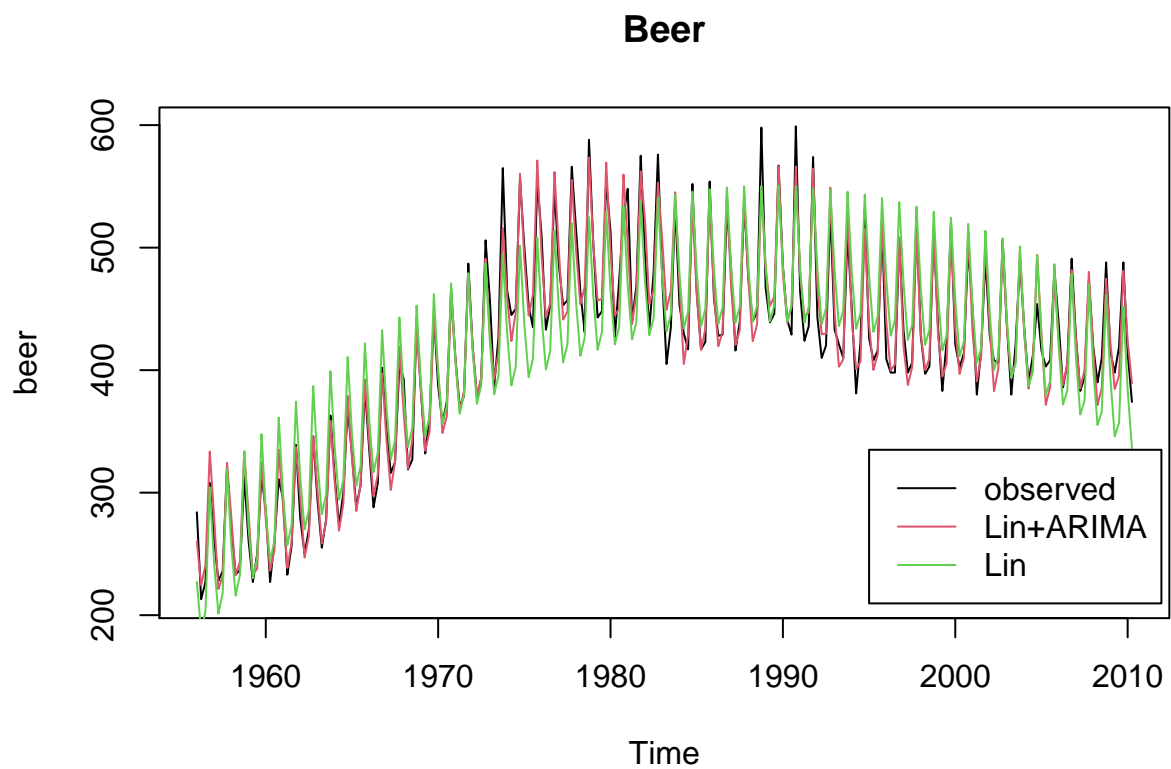
```
## Series: res.l2
## ARIMA(3,0,1)(1,0,1)[4] with zero mean
##
## Coefficients:
##      ar1      ar2      ar3      ma1      sar1      sma1
##    0.3671  0.3006  0.2271 -0.3429  0.8481 -0.5438
## s.e.  0.1637  0.0734  0.1042  0.1595  0.0666  0.1100
##
## sigma^2 estimated as 229.2:  log likelihood=-900.52
## AIC=1815.05   AICc=1815.58   BIC=1838.74
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.07967698 14.92971 11.53309 47.23239 165.2269 0.7952377
##              ACF1
## Training set 0.005866929
```

```
#Fitting on residuals and plot comparison
```

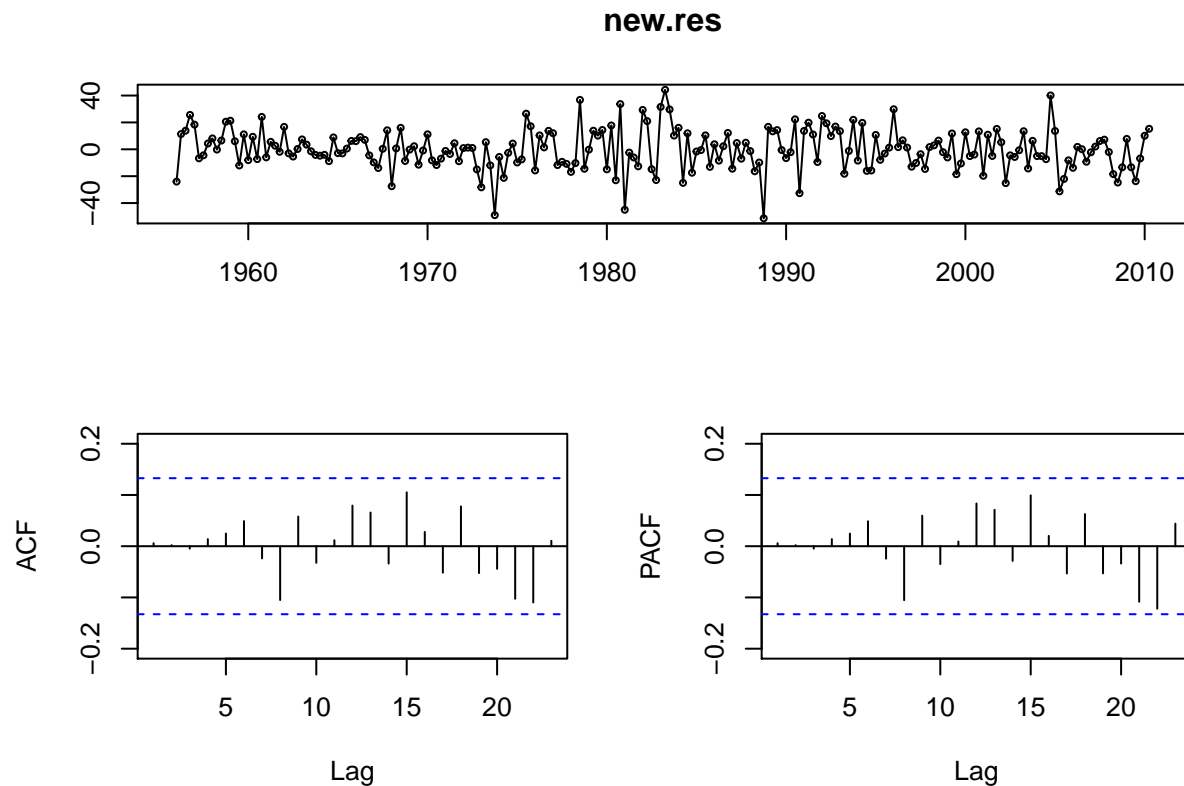
```
fit.ar <- fitted(arimaResiduals)
plot(fit.ar, lty=1)
lines(res.l2, lty=1,col=2)
legend(1990,63,legend= c("arima forecast", "observed t.s."), col=c(1,2), lty=2)
```



```
#Combination of tslm and ARIMA
fittbeer <- fit.l2 + fit.ar
plot(beer, type='l', main = "Beer")
lines(fittbeer, col=2)
lines(fit.l2,col=3)
legend(1995,335,legend= c("observed","Lin+ARIMA","Lin"), col=c(1,2,3), lty=1)
```



```
#New residuals  
new.res<-fittbeer-beer  
tsdisplay(new.res)
```

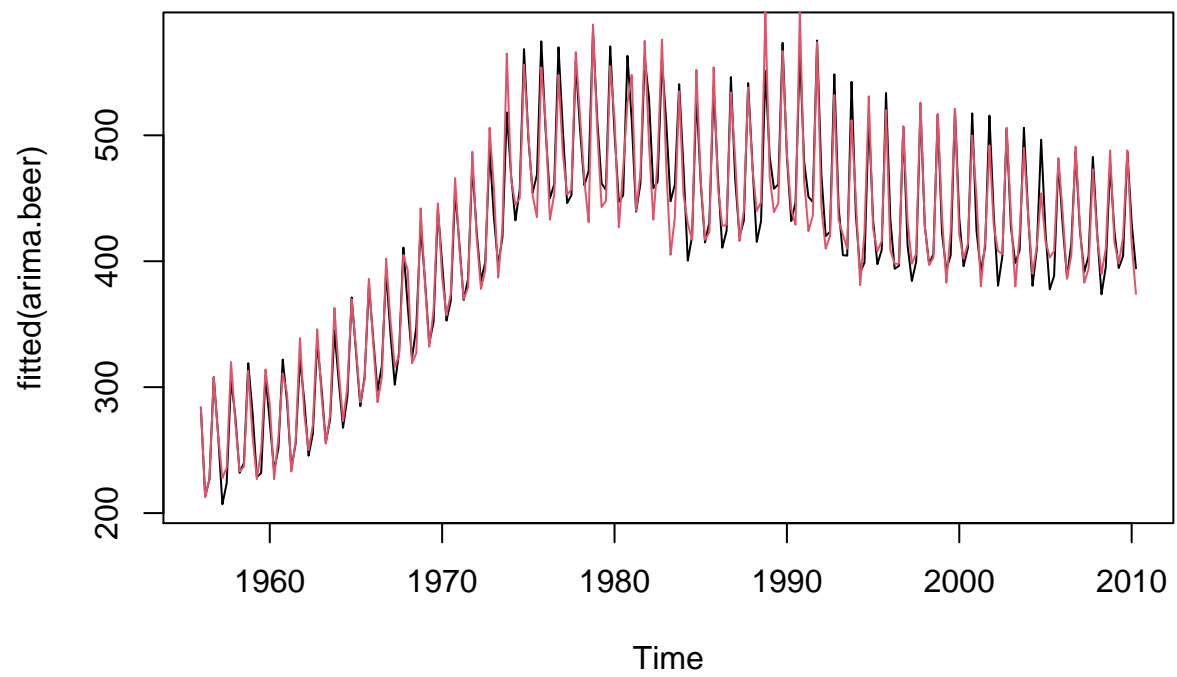



The model fits better than the previous and the residuals seem ok

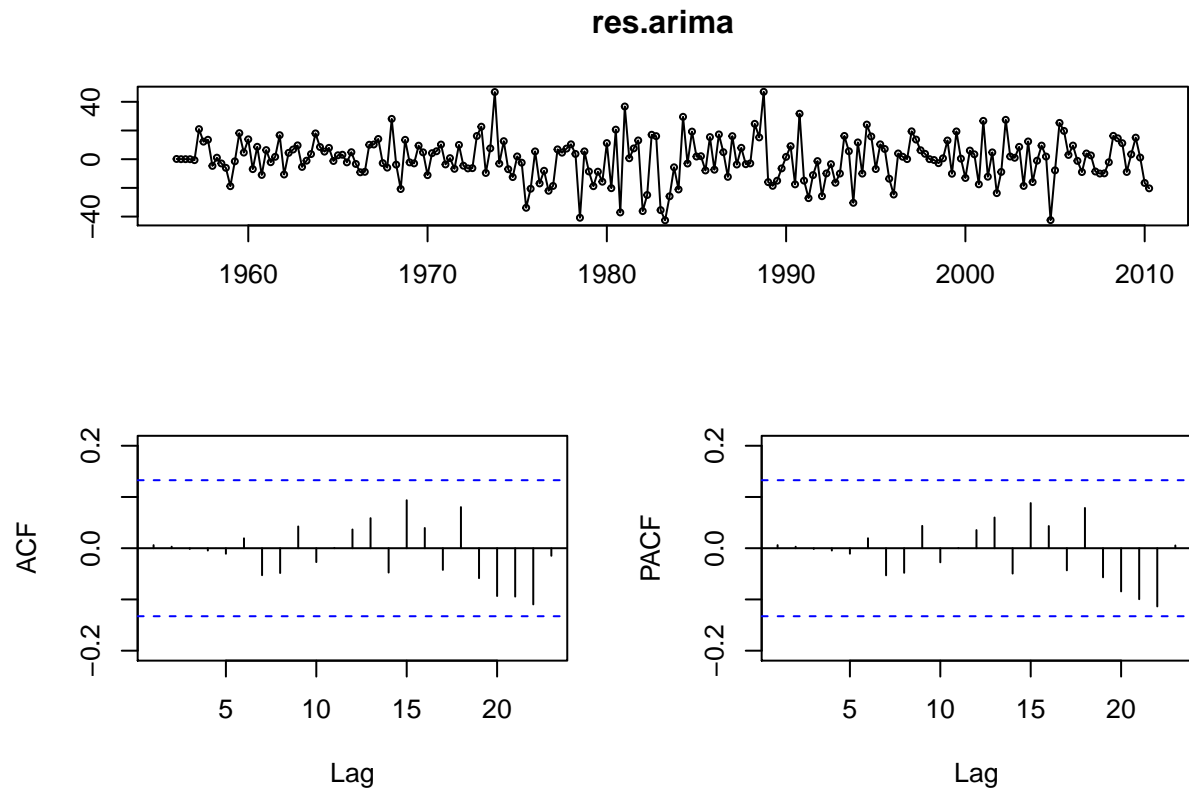
```
#ARIMA model (paramters of the best model evaluated using AIC)
arima.beer <- Arima(beer, order = c(2,1,1), seasonal = list(order=c(2,1,2), period=4))
arima.beer
```

```
## Series: beer
## ARIMA(2,1,1)(2,1,2)[4]
##
## Coefficients:
##          ar1      ar2      ma1      sar1      sar2      sma1      sma2
##      -0.5331 -0.2019 -0.422  -0.8447  0.0113  0.2214 -0.6996
## s.e.   0.1443   0.1178   0.137   0.1280  0.1213  0.1189   0.1203
##
## sigma^2 estimated as 236.4:  log likelihood=-883.29
## AIC=1782.59   AICc=1783.29   BIC=1809.48
```

```
#Fitting
plot(fitted(arima.beer))
lines(beer, col=2)
```



```
#Analysis of residuals  
res.arima<- residuals(arima.beer)  
tsdisplay(res.arima)
```

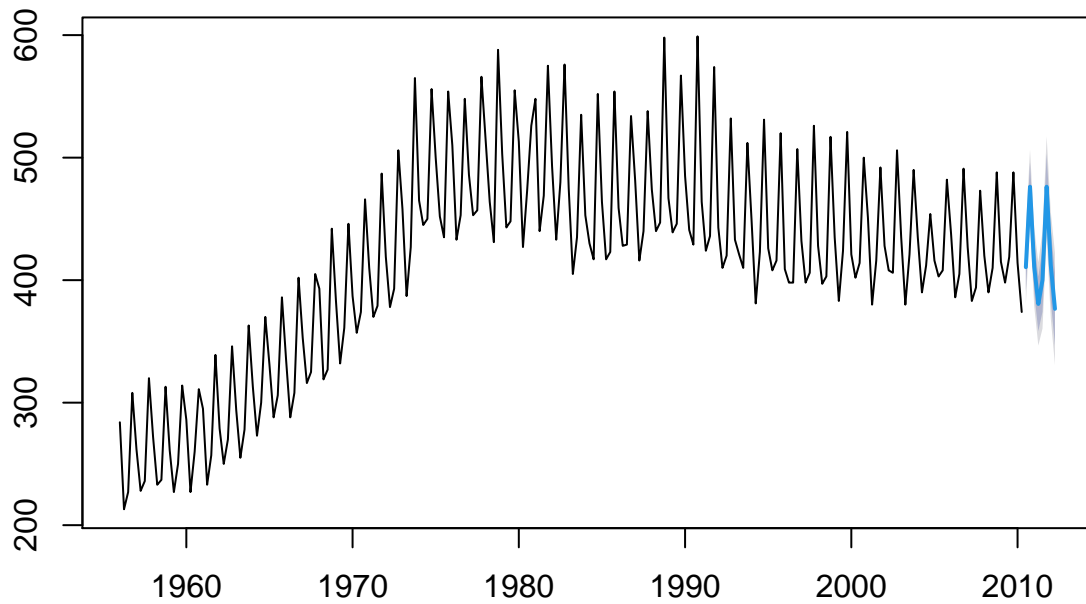


#This model has a good fit and the residuals are ok, but it has a complex structure and the parameter a

#Forecasts

```
forecast.arima <- forecast(arima.beer)
plot(forecast.arima)
```

Forecasts from ARIMA(2,1,1)(2,1,2)[4]



ARIMA models (part I)

Data on quarterly percentage change in US consumption, income, production, savings, unemployment

```
#All the data
uschange
```

##		Consumption	Income	Production	Savings	Unemployment
##	1970 Q1	0.61598622	0.97226104	-2.45270031	4.81031150	0.9
##	1970 Q2	0.46037569	1.16908472	-0.55152509	7.28799234	0.5
##	1970 Q3	0.87679142	1.55327055	-0.35870786	7.28901306	0.5
##	1970 Q4	-0.27424514	-0.25527238	-2.18545486	0.98522964	0.7
##	1971 Q1	1.89737076	1.98715363	1.90973412	3.65777061	-0.1
##	1971 Q2	0.91199291	1.44733417	0.90153584	6.05134180	-0.1
##	1971 Q3	0.79453885	0.53181193	0.30801942	-0.44583221	0.1
##	1971 Q4	1.64858747	1.16012514	2.29130441	-1.53087186	0.0
##	1972 Q1	1.31372218	0.45701150	4.14957387	-4.35859438	-0.2
##	1972 Q2	1.89147495	1.01662441	1.89062398	-5.05452579	-0.1
##	1972 Q3	1.53071400	1.90410126	1.27335290	5.80995904	-0.2
##	1972 Q4	2.31829471	3.89025866	3.43689207	16.04471706	-0.3
##	1973 Q1	1.81073916	0.70825266	2.79907636	-5.34886849	-0.3
##	1973 Q2	-0.04173996	0.79430954	0.81768862	8.42603436	0.0
##	1973 Q3	0.35423556	0.43381827	0.86899693	2.75879565	-0.1
##	1973 Q4	-0.29163216	1.09380979	1.47296187	11.14642986	0.1
##	1974 Q1	-0.87702794	-1.66168482	-0.88248358	-2.53351449	0.2

## 1974 Q2	0.35113555	-0.93835321	0.07427919	-6.59264464	0.3
## 1974 Q3	0.40959770	0.09448779	-0.41314971	0.51717884	0.5
## 1974 Q4	-1.47580863	-0.12259599	-4.06411893	11.34339540	1.3
## 1975 Q1	0.83225762	-0.16369546	-6.85103912	-5.47619069	1.4
## 1975 Q2	1.65583461	4.53650956	-1.33129558	24.30960536	0.2
## 1975 Q3	1.41942029	-1.46376532	2.42435972	-17.65616104	-0.4
## 1975 Q4	1.05437932	0.76166351	2.16904208	0.64809041	-0.2
## 1976 Q1	1.97998024	1.16825761	3.02720471	-2.95006644	-0.6
## 1976 Q2	0.91391607	0.51729906	1.27881101	-1.47455755	0.0
## 1976 Q3	1.05532326	0.73370026	1.30386487	-0.06754475	0.0
## 1976 Q4	1.29889825	0.59458339	1.77537765	-3.57672239	0.2
## 1977 Q1	1.13637586	-0.03108003	2.05516067	-9.16055658	-0.4
## 1977 Q2	0.54994073	1.23808955	3.05838507	9.09050404	-0.2
## 1977 Q3	0.94985262	1.51880293	1.10308888	7.94495719	-0.4
## 1977 Q4	1.49599724	1.91456240	0.63346850	6.69627648	-0.4
## 1978 Q1	0.57549599	0.70266687	-0.29339056	2.92296383	-0.1
## 1978 Q2	2.11120960	0.98314132	3.94815264	-6.81114259	-0.4
## 1978 Q3	0.41796279	0.71992620	0.87114701	4.79207162	0.1
## 1978 Q4	0.79792710	0.78553605	1.78447991	2.37118400	0.0
## 1979 Q1	0.50584598	1.05755946	0.42594327	7.77418337	-0.2
## 1979 Q2	-0.05775339	-0.86765105	-0.20491944	-5.28634896	-0.1
## 1979 Q3	0.97730010	0.47100340	-0.29723637	-1.84549644	0.2
## 1979 Q4	0.26826982	0.44037974	0.33560928	4.04959810	0.1
## 1980 Q1	-0.15391875	0.33827686	0.41056141	5.86168864	0.3
## 1980 Q2	-2.27411019	-1.46388507	-4.30076832	8.24322919	1.3
## 1980 Q3	1.07188123	1.21301507	-1.64181977	5.70775044	-0.1
## 1980 Q4	1.31644941	1.94243865	3.78045520	9.15098787	-0.3
## 1981 Q1	0.52472770	-0.26813406	0.24627687	-5.68139002	0.2
## 1981 Q2	-0.01728203	-0.02363025	0.30977573	0.88183993	0.1
## 1981 Q3	0.40165150	2.02680183	0.91707444	15.99035721	0.1
## 1981 Q4	-0.75287620	0.19560628	-2.25457797	7.80550650	0.9
## 1982 Q1	0.65938376	0.11969888	-2.07131293	-3.34243955	0.5
## 1982 Q2	0.36854173	0.57548997	-1.24766384	2.19400166	0.6
## 1982 Q3	0.76954464	0.53484410	-1.40050430	0.03499563	0.5
## 1982 Q4	1.80876006	0.44938311	-1.90375664	-9.57651468	0.7
## 1983 Q1	0.96802954	0.85588425	1.14655720	0.34595460	-0.5
## 1983 Q2	1.95946831	0.70632719	2.17942248	-10.17004699	-0.2
## 1983 Q3	1.73949442	1.49810999	3.36771897	0.21217916	-0.9
## 1983 Q4	1.56389332	2.13138911	2.58168445	8.21600068	-0.9
## 1984 Q1	0.84526442	2.02348788	2.89709545	13.86918150	-0.5
## 1984 Q2	1.41504495	1.64921136	1.53821324	4.38900229	-0.6
## 1984 Q3	0.76546608	1.36163845	0.72128740	6.51686089	0.1
## 1984 Q4	1.31380062	0.81927319	0.04115557	-2.87544931	0.0
## 1985 Q1	1.68655320	-0.23895759	0.32353159	-18.71008389	-0.1
## 1985 Q2	0.93436990	1.90677905	0.07020996	11.82871950	0.2
## 1985 Q3	1.90256675	-0.33536283	-0.14046924	-23.57393474	-0.3
## 1985 Q4	0.25656565	1.14181151	0.57978813	11.36628338	-0.1
## 1986 Q1	0.84304279	1.23951110	0.58132135	5.86126836	0.2
## 1986 Q2	1.11177390	1.31938549	-0.57641778	3.27551734	0.0
## 1986 Q3	1.79499406	0.70477150	0.37249329	-10.09044542	-0.2
## 1986 Q4	0.63768446	0.17977925	1.13734778	-4.82920131	-0.4
## 1987 Q1	0.01569397	0.81973366	1.30758228	12.46424452	0.0
## 1987 Q2	1.37731686	-0.97505791	1.75000563	-29.52866718	-0.4
## 1987 Q3	1.15225712	1.80185055	1.84366200	12.32810406	-0.3

## 1987 Q4	0.21016439	1.32743427	2.40645058	16.63076101	-0.2
## 1988 Q1	1.76316026	1.44861875	0.92013121	-0.96896505	0.0
## 1988 Q2	0.73053714	1.02084894	0.87316353	5.67776867	-0.3
## 1988 Q3	0.85083233	0.95820336	0.38103668	3.64649867	0.0
## 1988 Q4	1.13789838	0.96207024	0.70292025	-0.19730358	-0.1
## 1989 Q1	0.46064152	1.22693023	0.43372685	10.01461545	-0.3
## 1989 Q2	0.46937808	-0.29489091	-0.36675732	-8.15576525	0.3
## 1989 Q3	0.98950145	0.67822897	-0.62142121	-2.48622554	0.0
## 1989 Q4	0.43942767	0.80025832	0.42443392	5.44681102	0.1
## 1990 Q1	0.85543417	0.83939484	0.68265169	2.87544931	-0.2
## 1990 Q2	0.31230451	0.59572848	0.77446547	5.10951644	0.0
## 1990 Q3	0.40261313	0.03740765	0.41944800	-3.17767248	0.7
## 1990 Q4	-0.75910716	-0.79479735	-1.57345296	-0.17953326	0.4
## 1991 Q1	-0.34535008	0.21183290	-1.91422028	6.49315257	0.5
## 1991 Q2	0.83564224	0.69043356	0.59131506	-0.30920615	0.1
## 1991 Q3	0.48439843	0.36205181	1.36255645	-0.14086493	0.0
## 1991 Q4	-0.02626579	0.85100324	0.21710308	11.34193010	0.4
## 1992 Q1	1.85996999	2.12421067	-0.13365365	7.23265150	0.1
## 1992 Q2	0.68354371	1.04095059	1.76874773	5.46708666	0.4
## 1992 Q3	1.07661214	0.43562041	0.76167388	-5.93646090	-0.2
## 1992 Q4	1.18372396	0.34210852	1.05024577	-5.88618856	-0.2
## 1993 Q1	0.37817936	0.55877186	0.87901471	2.63464703	-0.4
## 1993 Q2	0.89392729	0.17627103	0.21755108	-6.91664675	0.0
## 1993 Q3	1.09813766	0.05868803	0.40135891	-11.99337844	-0.3
## 1993 Q4	0.88122025	0.65496353	1.49618275	-1.83708870	-0.2
## 1994 Q1	1.14064791	0.69846579	1.22213656	-5.18600629	0.0
## 1994 Q2	0.77176225	1.05367166	1.78250275	5.15609751	-0.4
## 1994 Q3	0.77214364	0.59247377	1.26718100	-2.42215898	-0.2
## 1994 Q4	1.07014805	1.38110661	2.04370404	6.32351898	-0.4
## 1995 Q1	0.26420505	0.94873528	1.02552601	10.11514398	-0.1
## 1995 Q2	0.89311141	0.22780635	0.33785685	-10.60541172	0.2
## 1995 Q3	0.91264702	0.88957006	0.90043887	-0.11570727	0.0
## 1995 Q4	0.70025425	0.57591998	0.87467273	-2.90726686	0.0
## 1996 Q1	0.92360967	0.95255663	0.69285195	2.55933958	-0.1
## 1996 Q2	1.07997887	0.95161791	2.11134752	-0.75802112	-0.2
## 1996 Q3	0.60055799	0.79369738	1.24418680	3.33843952	-0.1
## 1996 Q4	0.78298122	0.52035746	1.35396890	-3.33843952	0.2
## 1997 Q1	1.04949253	0.99858552	1.86714700	0.61269338	-0.2
## 1997 Q2	0.45219855	0.85103564	1.48763922	6.17532322	-0.2
## 1997 Q3	1.69654264	1.18352222	2.28632066	-7.22796452	-0.1
## 1997 Q4	1.18062797	1.42325742	2.48091341	5.43456565	-0.2
## 1998 Q1	1.02693626	2.10753052	1.10343775	19.35335228	0.0
## 1998 Q2	1.75069399	1.38767133	0.65122238	-4.81709478	-0.2
## 1998 Q3	1.30596977	1.01464427	0.72551955	-3.12983982	0.1
## 1998 Q4	1.45888615	0.80893032	1.44421674	-9.14923404	-0.2
## 1999 Q1	0.94821191	0.89173174	1.10341663	1.88735718	-0.2
## 1999 Q2	1.46971415	0.24722185	0.98574261	-23.49652903	0.1
## 1999 Q3	1.12921436	0.66729226	0.90279881	-9.86264835	-0.1
## 1999 Q4	1.45748895	1.46092242	1.75533234	2.35825225	-0.2
## 2000 Q1	1.51106759	1.95061335	0.99682019	12.28684080	0.0
## 2000 Q2	0.95508878	1.03174349	1.23293805	1.28001748	0.0
## 2000 Q3	0.96797647	1.16178668	-0.10225268	2.57390229	-0.1
## 2000 Q4	0.88629738	0.33725343	-0.20388383	-13.16296208	0.0
## 2001 Q1	0.42159086	0.84865826	-1.35143911	13.22491995	0.4

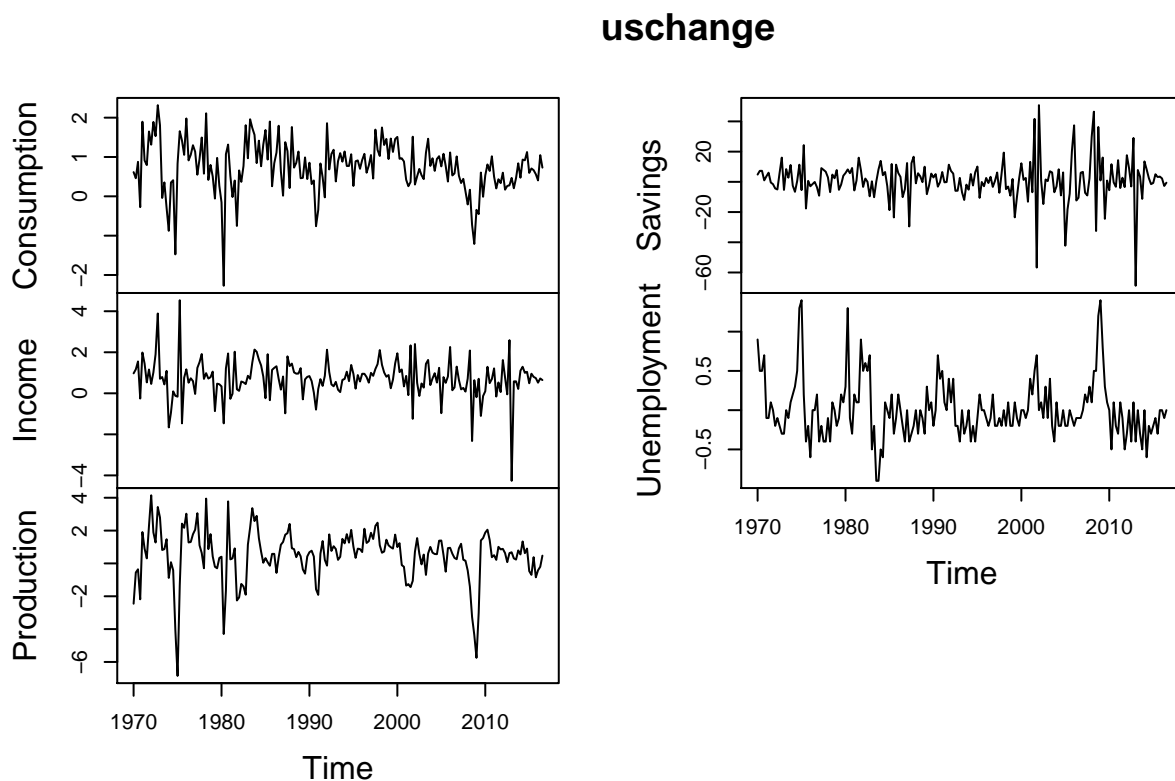
##	2001	Q2	0.25689982	-0.08818148	-1.25954437	-6.89043916	0.2
##	2001	Q3	0.36381084	2.33678920	-1.44101744	41.66826457	0.5
##	2001	Q4	1.51630321	-1.24443353	-1.06013675	-56.75209674	0.7
##	2002	Q1	0.29958257	2.40331419	0.70916406	50.75796205	0.0
##	2002	Q2	0.50899032	0.50559877	1.54280957	0.87861837	0.1
##	2002	Q3	0.69667241	-0.12828194	0.59478143	-14.70397426	-0.1
##	2002	Q4	0.53634306	0.47941927	-0.05776556	1.58733492	0.3
##	2003	Q1	0.43826169	0.27834026	0.53922789	0.49744834	-0.1
##	2003	Q2	1.10719086	1.43729445	-0.69876172	7.00891625	0.4
##	2003	Q3	1.46377882	1.62544947	0.60727351	6.18413150	-0.2
##	2003	Q4	0.77334046	0.40353864	1.00599126	-6.89274778	-0.4
##	2004	Q1	0.96768535	0.72653162	0.65792806	-2.96152040	0.1
##	2004	Q2	0.64760607	0.98056746	0.57461780	8.30885627	-0.2
##	2004	Q3	0.95117167	0.52450113	0.56330030	-8.99318286	-0.2
##	2004	Q4	1.02041702	1.24238706	1.38522763	6.23585017	0.0
##	2005	Q1	0.76172556	-0.96827007	1.39435718	-42.28191228	-0.2
##	2005	Q2	1.08136588	0.78835467	0.50586367	-18.27592893	-0.2
##	2005	Q3	0.77186494	0.51136949	-0.50305848	-7.87665229	0.0
##	2005	Q4	0.37591485	0.82191843	0.93365010	20.37236078	-0.1
##	2006	Q1	1.11522822	2.25904474	0.95057853	37.40653542	-0.2
##	2006	Q2	0.53100554	0.14987813	0.59636010	-12.34810568	-0.1
##	2006	Q3	0.58208747	0.28490722	0.33552773	-10.55276140	-0.1
##	2006	Q4	1.01434389	1.30059162	0.25603401	6.03100080	-0.1
##	2007	Q1	0.52486184	0.65373993	0.91794957	6.60516929	0.0
##	2007	Q2	0.33874119	0.19260870	1.19594247	-7.23648452	0.2
##	2007	Q3	0.44391875	0.26238732	0.22356909	-9.00674555	0.1
##	2007	Q4	0.12505584	0.08392938	0.16424632	2.32887238	0.3
##	2008	Q1	-0.20652548	0.71926565	-0.42872571	29.83728599	0.1
##	2008	Q2	0.16783443	2.08693775	-1.41297022	46.43989041	0.5
##	2008	Q3	-0.72499446	-2.32611860	-3.26349945	-32.53252494	0.5
##	2008	Q4	-1.21068558	0.64019534	-4.35417741	36.31240490	1.2
##	2009	Q1	-0.34354370	-0.18888849	-5.75045075	0.92306020	1.4
##	2009	Q2	-0.45174364	0.70899368	-3.00372447	16.09059408	0.8
##	2009	Q3	0.60491332	-1.10343180	1.39880419	-24.49229966	0.3
##	2009	Q4	-0.01115014	-0.13213193	1.54400617	0.84829220	0.1
##	2010	Q1	0.53481740	0.10094986	1.88006931	-5.54399051	0.0
##	2010	Q2	0.81040406	1.29229259	2.05402479	11.65612884	-0.5
##	2010	Q3	0.64501881	0.49678098	1.42683671	-0.35208609	0.1
##	2010	Q4	1.01833874	0.69495229	0.37927209	-3.27335958	-0.2
##	2011	Q1	0.50041315	1.21571502	0.50174040	14.33860193	-0.3
##	2011	Q2	0.20141978	-0.15658108	0.21878696	-4.07705131	0.1
##	2011	Q3	0.43372599	0.52891255	1.01113866	2.72250400	-0.1
##	2011	Q4	0.33593895	0.06074719	0.85151692	-3.45447712	-0.5
##	2012	Q1	0.60108995	1.62204885	0.88651817	17.62530510	-0.3
##	2012	Q2	0.16942956	0.76689543	0.62923586	8.96949710	0.0
##	2012	Q3	0.26416034	-0.05071452	0.07880166	-3.04922177	-0.4
##	2012	Q4	0.27877186	2.59106697	0.63305509	29.04670355	0.1
##	2013	Q1	0.46861292	-4.26525047	0.67713243	-68.78826698	-0.4
##	2013	Q2	0.20545802	0.58146541	0.30744961	7.81647729	0.0
##	2013	Q3	0.46641787	0.58328912	0.23440888	3.49400682	-0.3
##	2013	Q4	0.83917367	0.21494896	0.79208722	-11.27661450	-0.5
##	2014	Q1	0.47345118	1.10369487	0.54709166	13.52020248	0.0
##	2014	Q2	0.93375698	1.29390492	1.33801074	8.24404770	-0.6
##	2014	Q3	0.91687178	0.99853396	0.62352731	2.46195256	-0.2

```
## 2014 Q4  1.12533250  1.04641801  0.90355427 -1.51305022      -0.3
## 2015 Q1  0.59624005  0.49040680 -0.46710878 -0.75840017      -0.2
## 2015 Q2  0.70814389  0.95495949 -0.69702162  5.02391773      -0.1
## 2015 Q3  0.66496956  0.80166267  0.38060610  3.18092976      -0.3
## 2015 Q4  0.56167978  0.74006260 -0.84554638  3.48278601       0.0
## 2016 Q1  0.40468216  0.51902540 -0.41793048  2.23653405       0.0
## 2016 Q2  1.04770741  0.72372078 -0.20331883 -2.72150106      -0.1
## 2016 Q3  0.72959779  0.64470081  0.47491844 -0.57285793       0.0
```

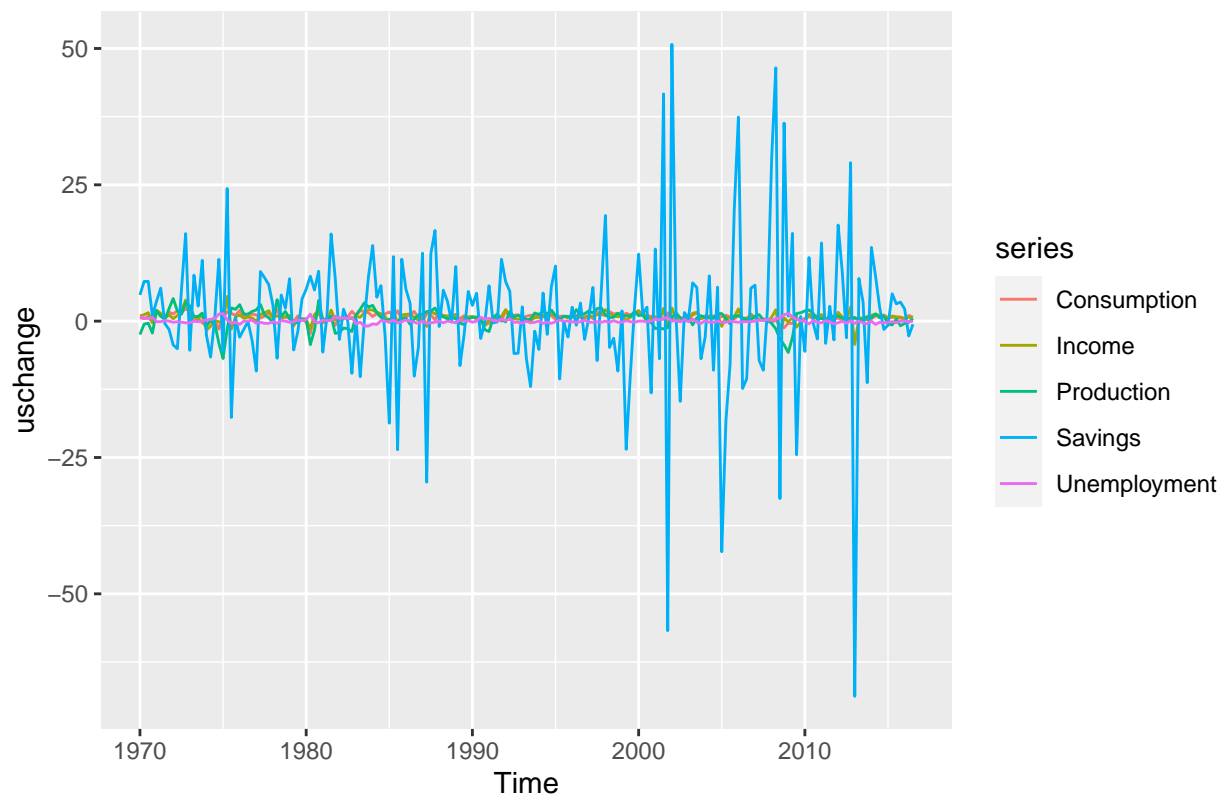
```
str(uschange)
```

```
## Time-Series [1:187, 1:5] from 1970 to 2016: 0.616 0.46 0.877 -0.274 1.897 ...
## - attr(*, "dimnames")=List of 2
## ..$ : NULL
## ..$ : chr [1:5] "Consumption" "Income" "Production" "Savings" ...
```

```
plot(uschange)
```



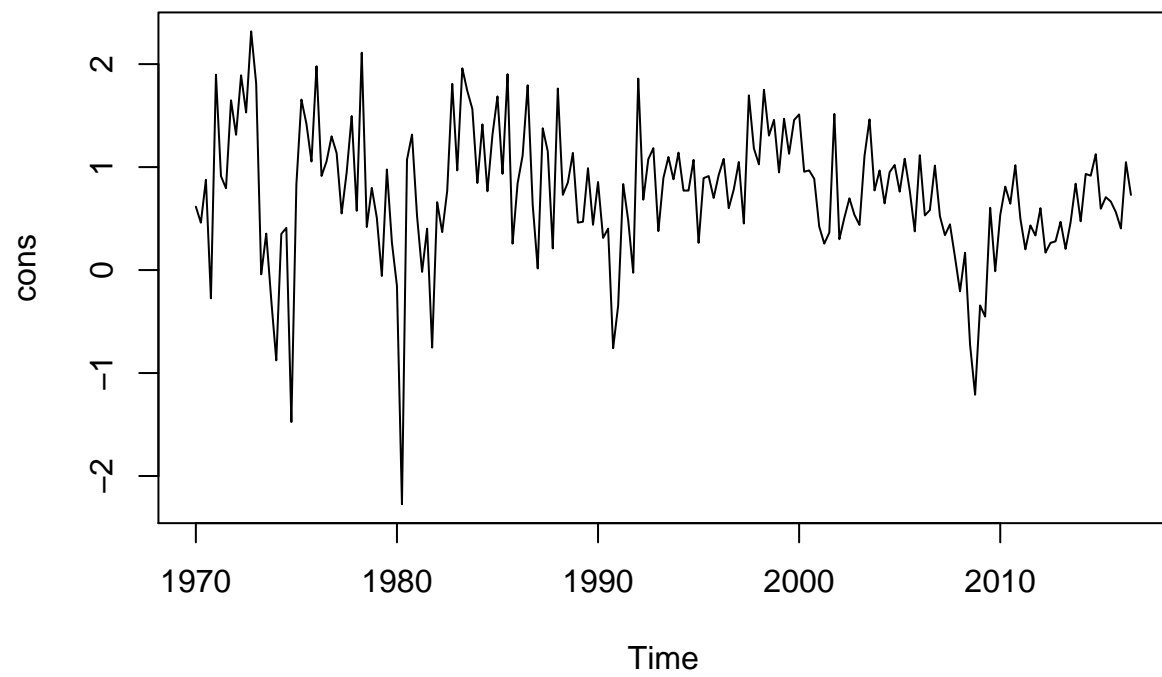
```
#Unique visualization of the series, to find possible relations
autoplot(uschange)
```

Exercise 2: try to study the behavior of these series together

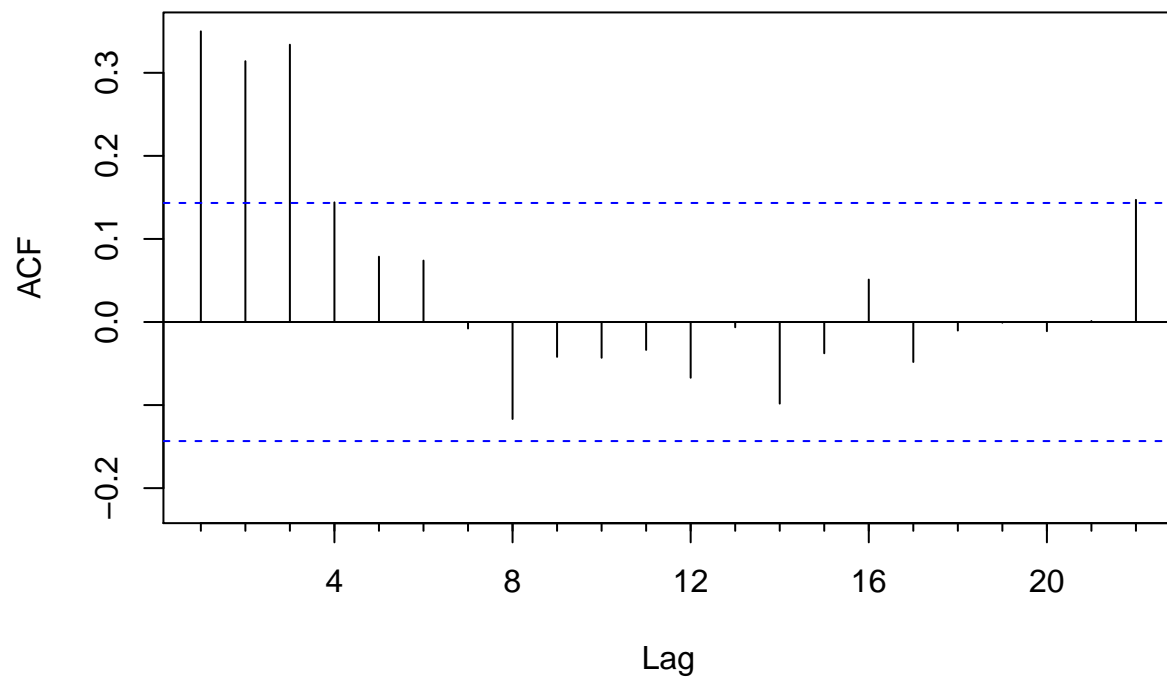
#Using a multiple regression model, for example studying the consumption against the other variables in this way we can see if there are significant relations between these variables. `molde<-tslm(uschange[,1]~ uschange[,2] + uschange[,3] + uschange[,4] + uschange[,5])` We will consider only the series of consumption.

```
#Data exploration
cons<- uschange[,1]
plot(cons)
```



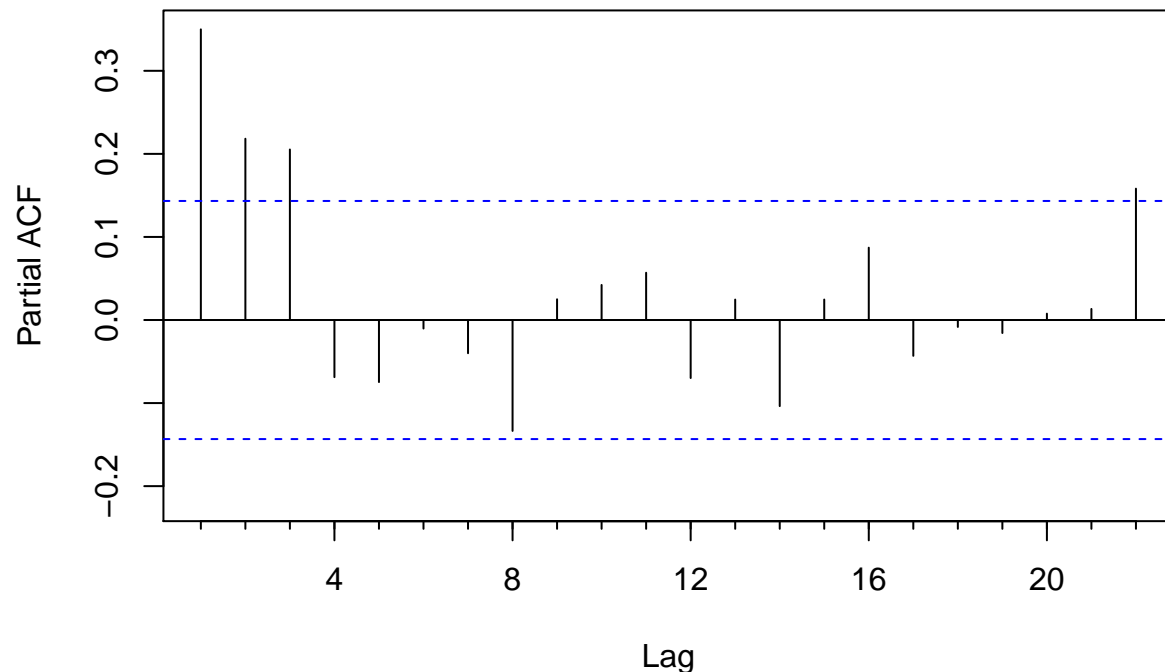
Acf(cons)

Series cons



Pacf(cons)

Series cons



We apparently not dealing with a strong trend, this is due to the fact that by construction this series can't have a strong trend (this dataset is related to the percentage change in these variables). We are not sure of if there is a seasonality pattern. This indicate that there is NOT a seasonal behavior (even if we have quarterly data). Same result of ACF, only the first lags are significant.

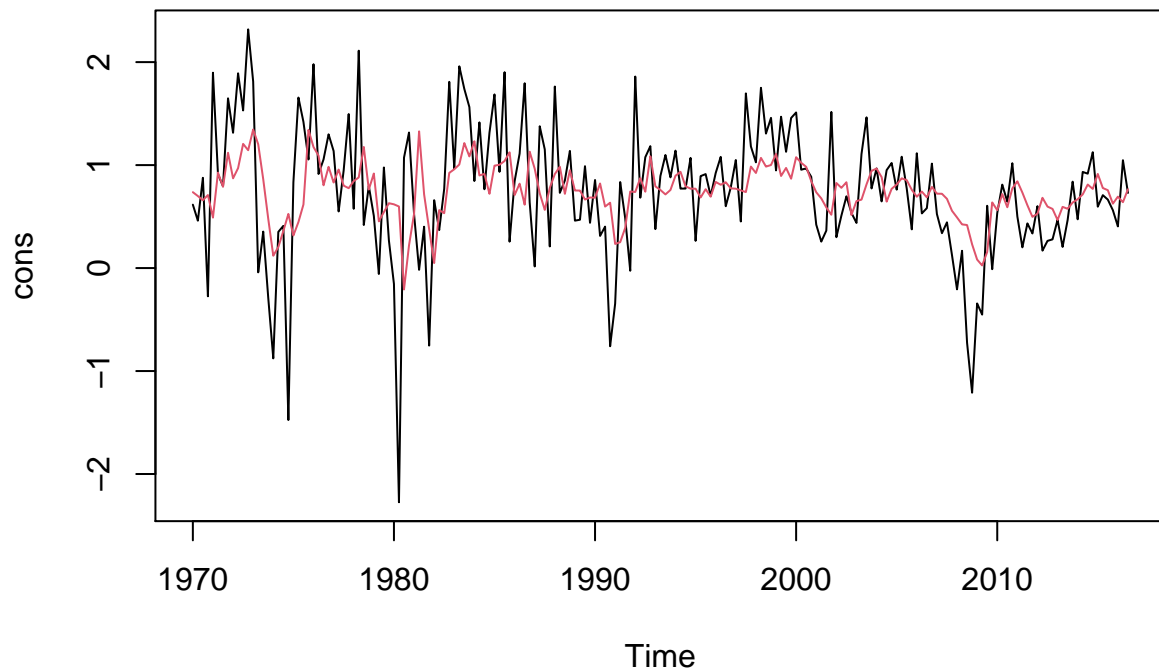
General indication: if the ACF is exponentially decaying and there is a significant spike at lag p (LAST lag and nothing else after) in PACF, it may be an $ARMA(p,d,0)$. If the PACF is exponentially decaying and there is a significant spike at lag p in ACF, it may be an $ARMA(0,d,q)$. In our case is difficult to see a clear situation between the previous general ones. We have to apply a trial and error approach based on the previous general rules. In our case it seems similar to the second situation: PACF has an exp. decaying and a spike in lag 3 in ACF (after we have any spikes). $d=0$ beacuse we have a stationary t.s..

```
arima1<- Arima(cons, order=c(0,0,3))
fitted(arima1)
```

##		Qtr1	Qtr2	Qtr3	Qtr4
## 1970		0.73705424	0.69967966	0.65866101	0.70938625
## 1971		0.49004929	0.92704762	0.78984511	1.11956181
## 1972		0.87155432	0.97056317	1.20620576	1.14445402
## 1973		1.34571585	1.20217877	0.86290396	0.47701083
## 1974		0.11990901	0.20414005	0.35980715	0.52574620
## 1975		0.31643226	0.44685913	0.61720980	1.34187730
## 1976		1.17581151	1.09144588	0.80387699	0.98320487
## 1977		0.83083538	0.95675929	0.80049400	0.77566256
## 1978		0.84463862	0.87994825	1.17626296	0.76260862
## 1979		0.91809555	0.45390041	0.54363676	0.62977278

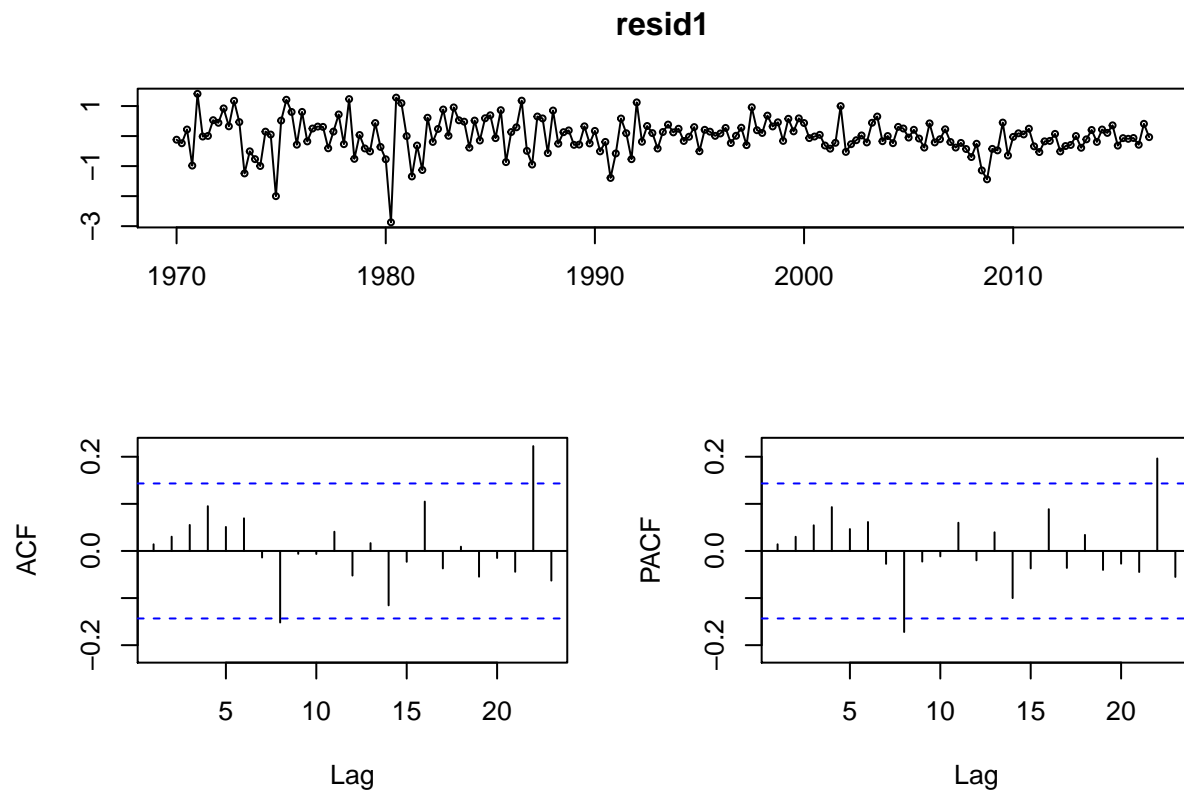
```
## 1980 0.61892320 0.59812702 -0.20815929 0.22088353
## 1981 0.52504023 1.32789938 0.71596915 0.37756282
## 1982 0.04848762 0.56315206 0.53288176 0.92441387
## 1983 0.95968741 1.00575039 1.21395589 1.08434770
## 1984 1.23159810 0.89938442 0.91452658 0.72129429
## 1985 0.99449733 1.00343081 1.03993225 1.12390311
## 1986 0.70912365 0.81971220 0.61563610 1.13023430
## 1987 0.96466593 0.72586720 0.56507988 0.77795562
## 1988 0.91287170 0.98393578 0.72103814 0.94967494
## 1989 0.75338857 0.75271150 0.66537484 0.68522325
## 1990 0.68361781 0.82122091 0.59709404 0.63508456
## 1991 0.23416754 0.25138109 0.38944774 0.74344527
## 1992 0.73881935 0.87369649 0.74165324 1.08498174
## 1993 0.79360328 0.75834173 0.71536312 0.75822465
## 1994 0.89669026 0.93482368 0.79424757 0.77135255
## 1995 0.77081624 0.68502340 0.76615600 0.69300376
## 1996 0.83653392 0.80885344 0.83342626 0.77384279
## 1997 0.77083529 0.75420647 0.73810775 0.98582185
## 1998 0.92320728 1.07023569 0.98540295 1.00076507
## 1999 1.10881183 0.89431754 0.97252966 0.86797521
## 2000 1.07655122 1.02237084 0.98324860 0.84471803
## 2001 0.73602968 0.67677637 0.58874478 0.51765762
## 2002 0.82618877 0.77919732 0.83335758 0.51504967
## 2003 0.65053097 0.66453539 0.81292902 0.94391928
## 2004 0.96659652 0.88371173 0.64535793 0.76945235
## 2005 0.81156081 0.87170385 0.85366671 0.76021699
## 2006 0.69295235 0.74294229 0.68630822 0.78845330
## 2007 0.72231846 0.72148306 0.67236281 0.55610793
## 2008 0.49178840 0.42438616 0.41810181 0.23045714
## 2009 0.08271295 0.02514980 0.15547323 0.63743167
## 2010 0.56265653 0.71856833 0.59045046 0.77308180
## 2011 0.84264309 0.73324621 0.61004536 0.49745391
## 2012 0.52822192 0.68251361 0.59691601 0.57458039
## 2013 0.46674195 0.59440201 0.57543318 0.63656576
## 2014 0.66850614 0.71569146 0.81104586 0.76843909
## 2015 0.91431334 0.77712032 0.75629064 0.62552150
## 2016 0.69361865 0.63958689 0.76517975
```

```
#Plot. It doesn't fit in a good way: it doesn't capture the strong pos. and neg. spikes
plot(cons)
lines(fitted(arima1), col=2)
```



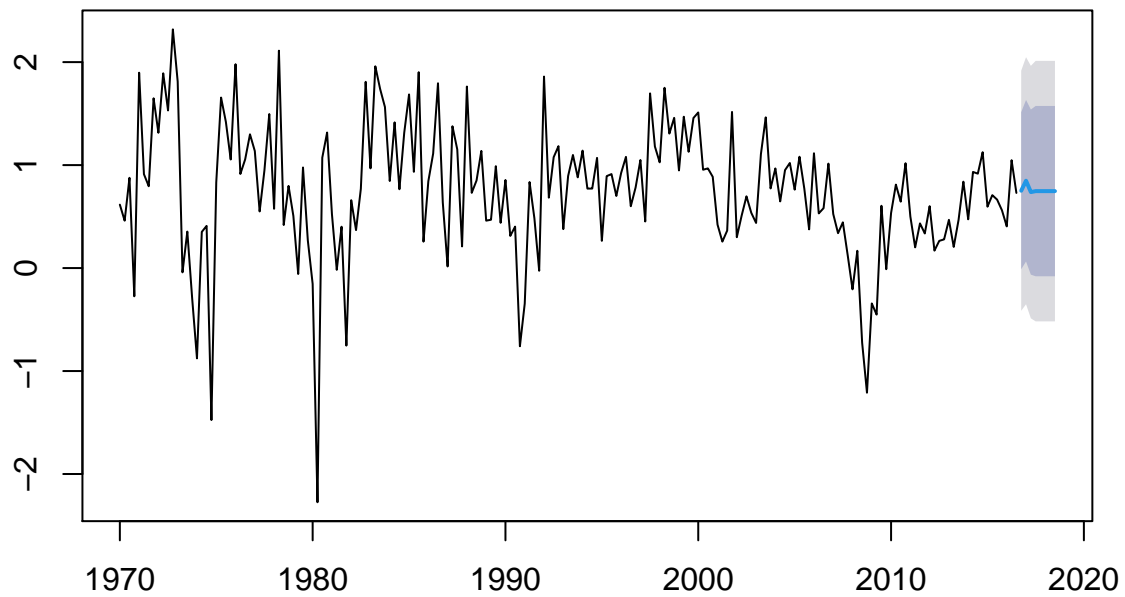
Residuals (we try in any case to calculate the residuals plots, although we have seen that the fitted values of the model aren't good). In a common case (see the guide), after the fitting (and the comparison with other model with AIC) we see directly the residual plot to see the presence of white noise. In this case we see the fitted values instead residuals. In ACF and PACF we have a not so easily understandable significant lag in position 22. The rest is globally acceptable

```
resid1<- residuals(arima1)
tsdisplay(resid1)
```



```
#Forecasting (although is not a good model)
for1<- forecast(arima1)
#In any case an ARIMA model performs, in terms of forecasting, a constant behavior. This is a common
#problematic feature of ARIMA models: if we don't have a seasonality, the forecasting has a constant
#behavior (ARIMAs are good describers of t.s., but bad forecaster)
plot(for1)
```

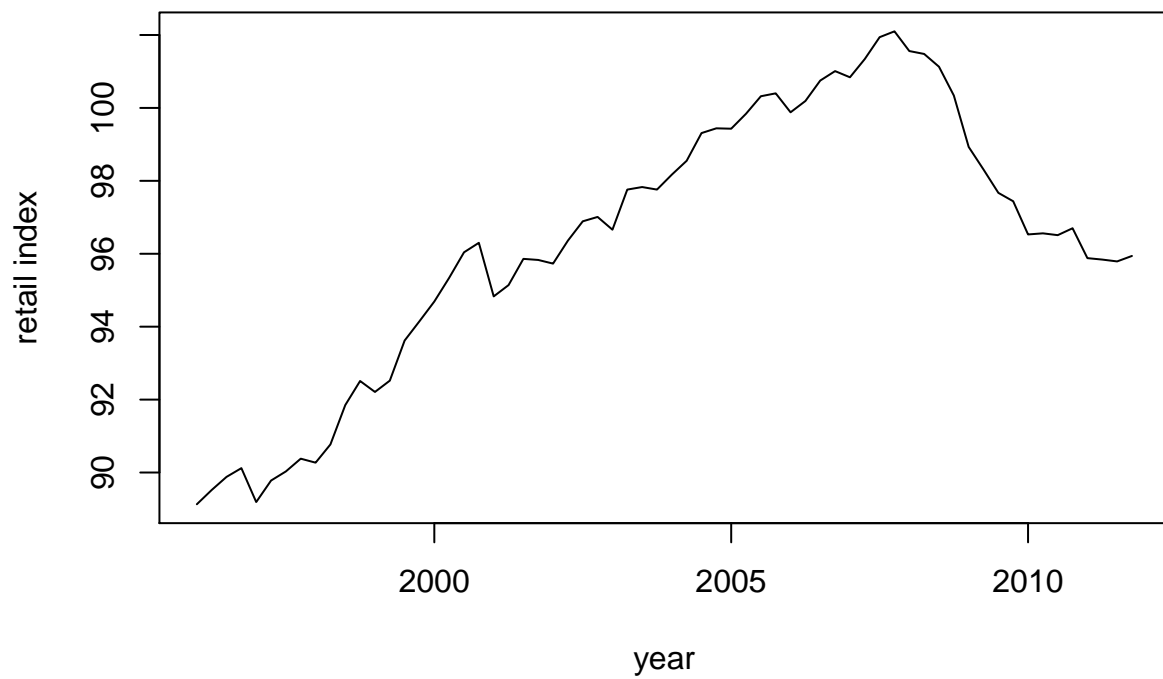
Forecasts from ARIMA(0,0,3) with non-zero mean



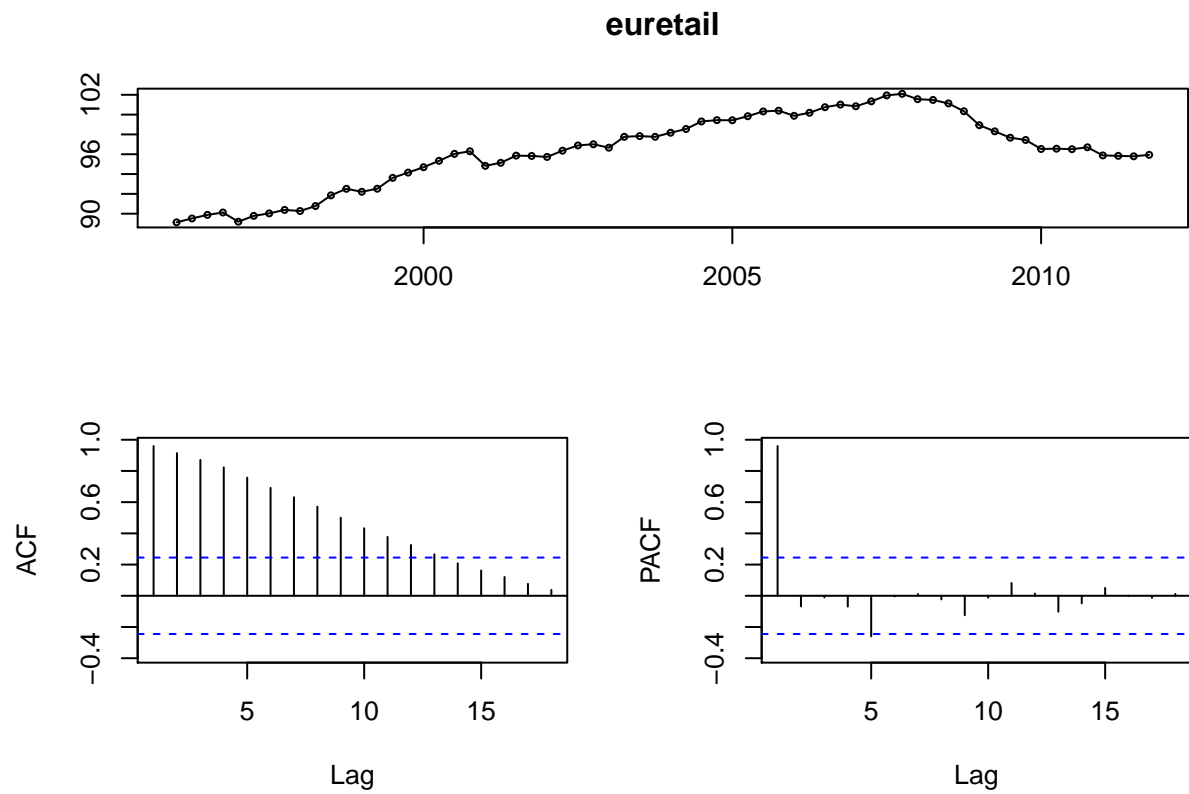
ARIMA models (part II) (Not treated during class The sequent argument is ARMAX)

Data on retail trade index in Euro area (1996-2011)

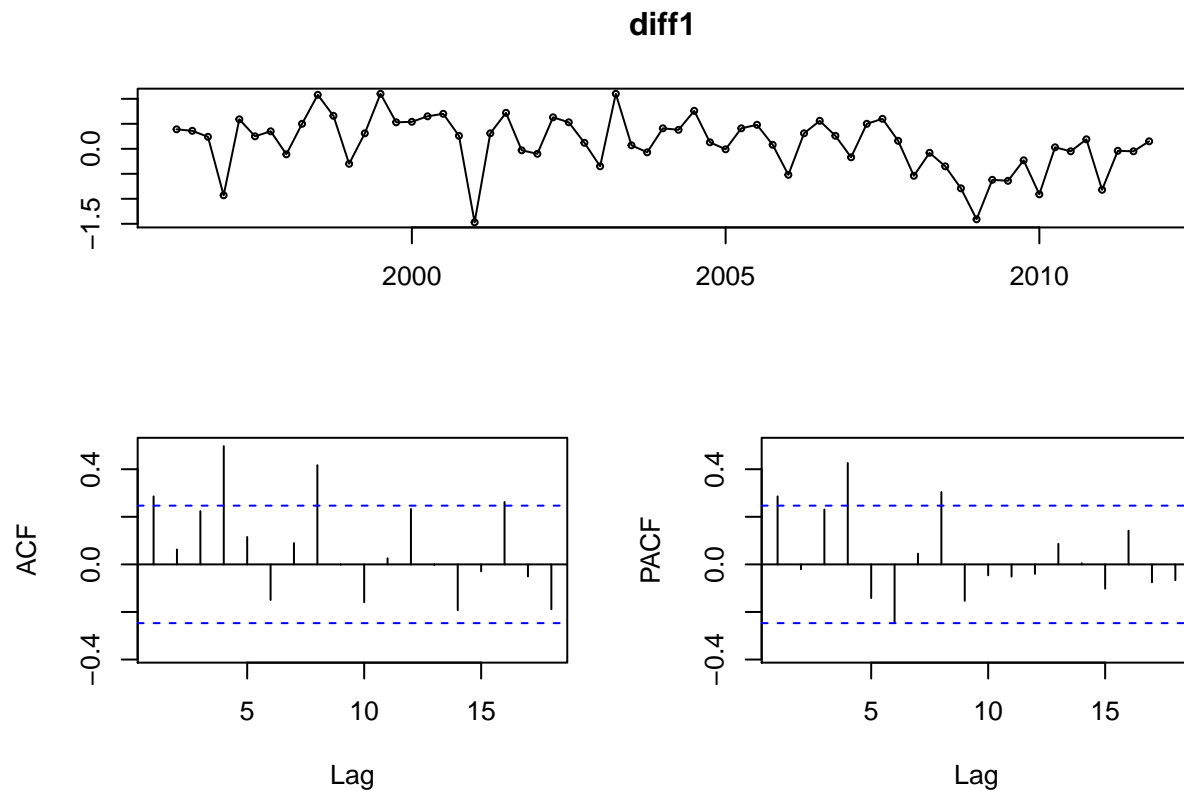
```
plot(euretail, ylab="retail index", xlab="year")
```

```
tsdisplay(euretail)
```



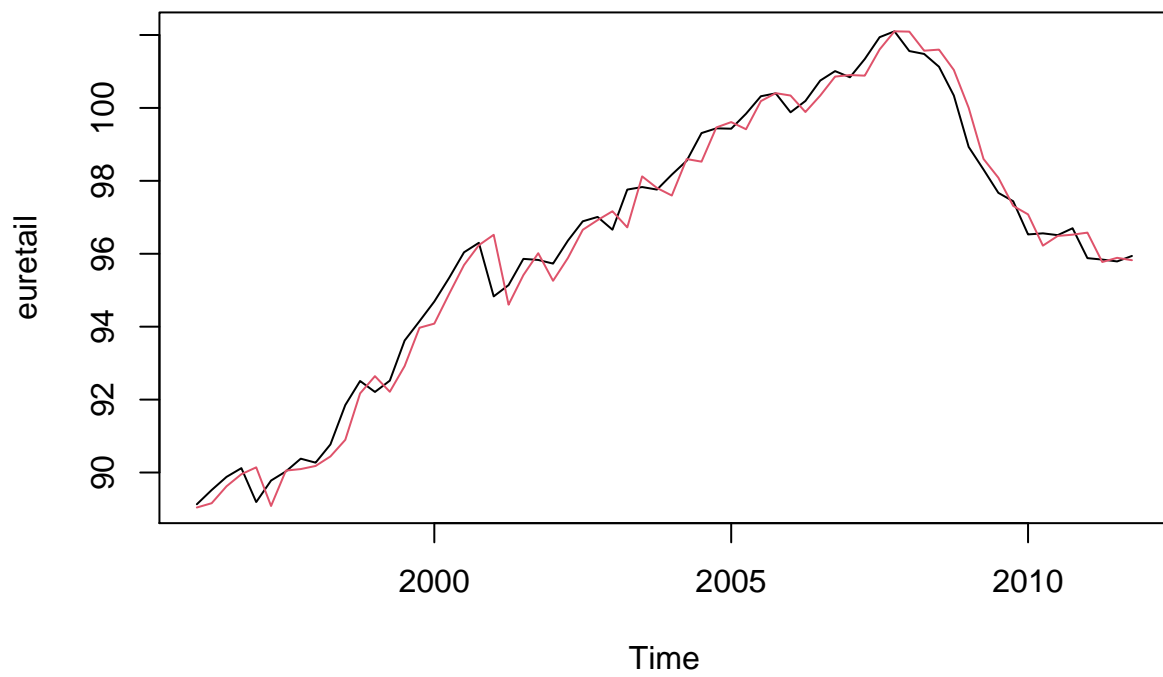
```
#Differencing  
diff1<- diff(euretail)  
tsdisplay(diff1)
```



Now we will proceed with a stepwise approach that modifies the parameters one at a time

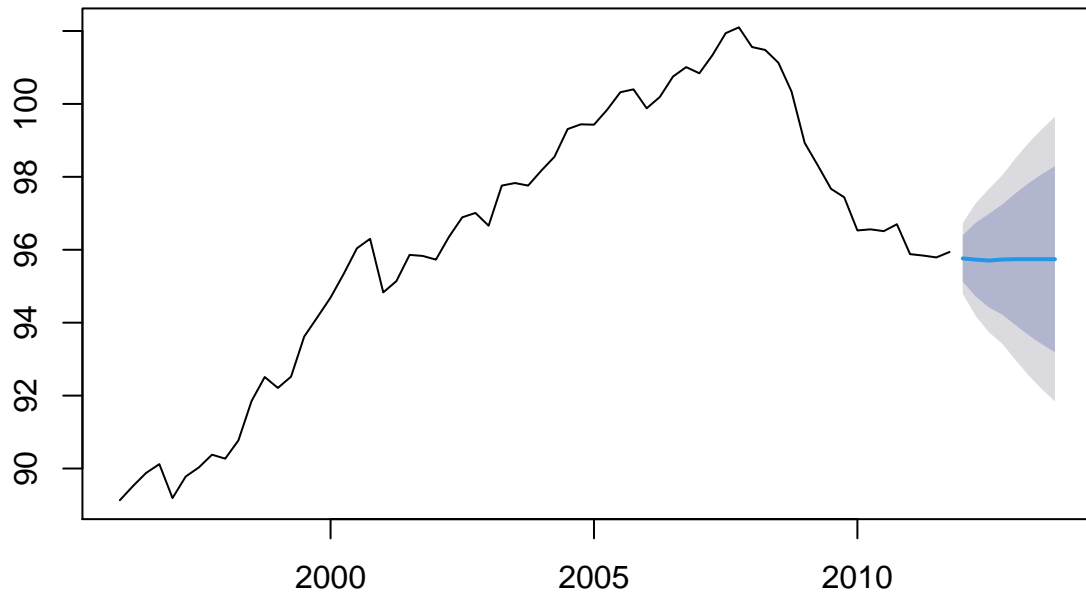
```
#We fit the first Arima model
a1<- Arima(euretail, order=c(0,1,1), seasonal=c(0,0,1))
fit1<- fitted(a1)

plot(euretail)
lines(fit1, col=2)
```

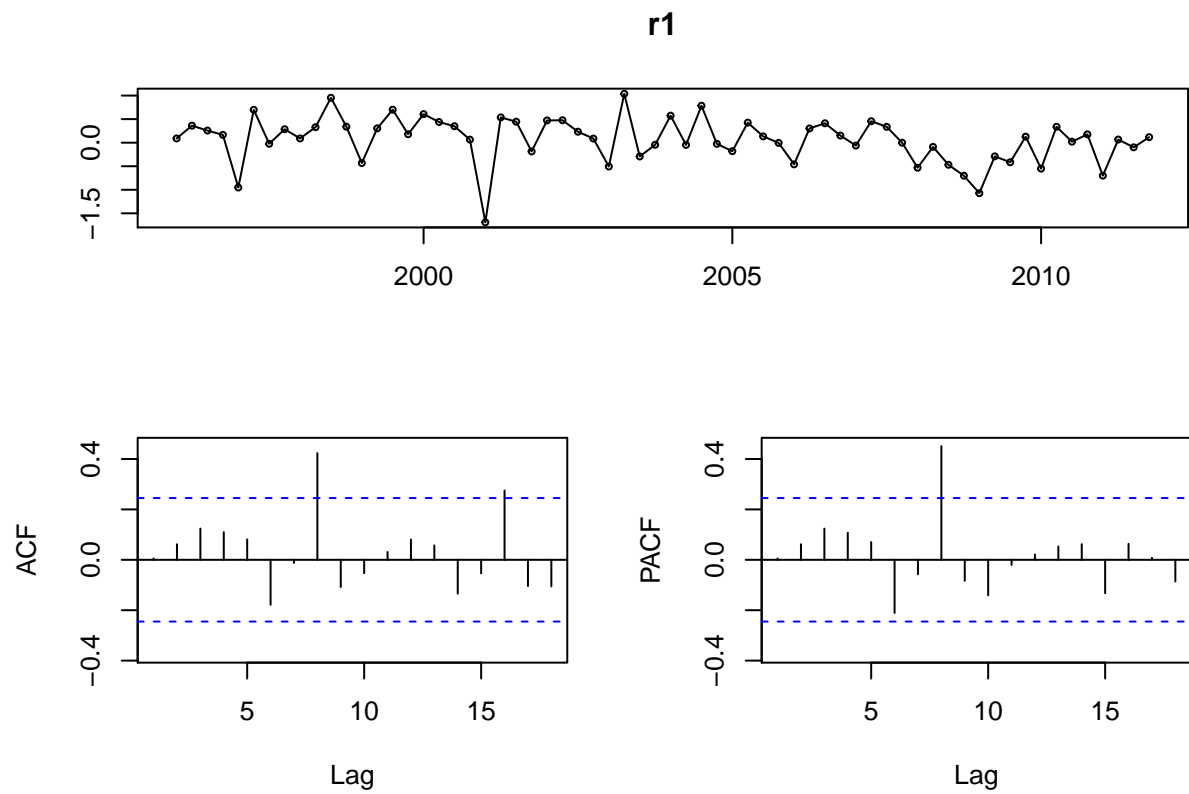


```
f1<- forecast(a1)  
plot(f1)
```

Forecasts from ARIMA(0,1,1)(0,0,1)[4]

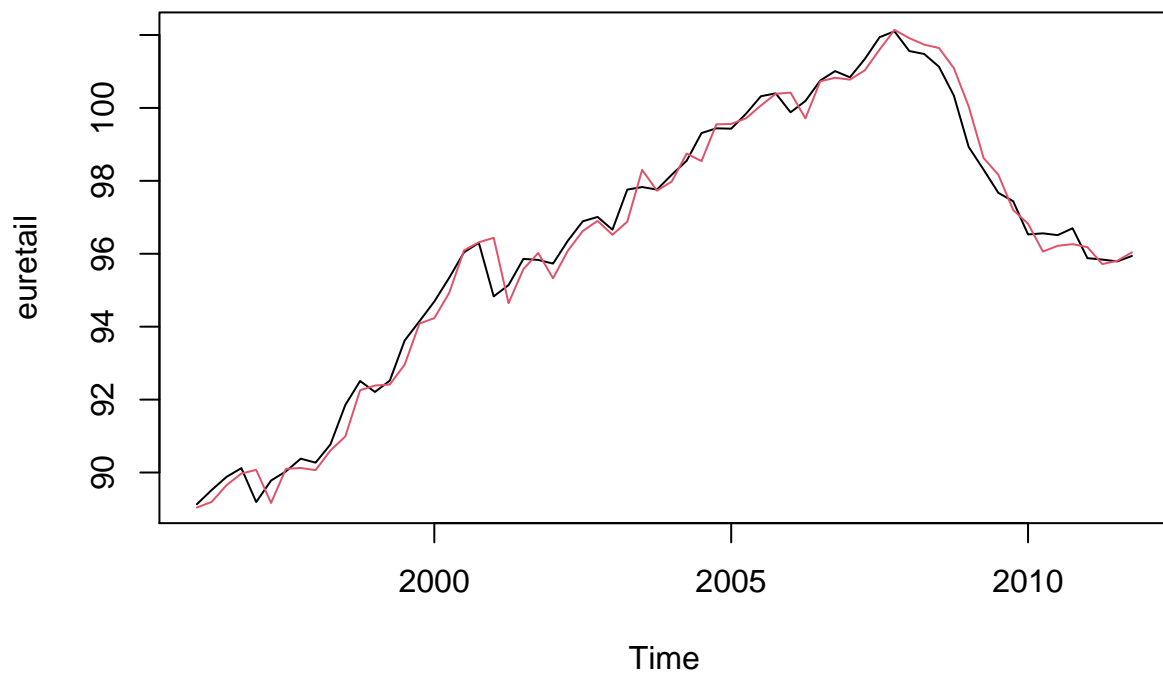


```
r1<- residuals(a1)
tsdisplay(r1)
```



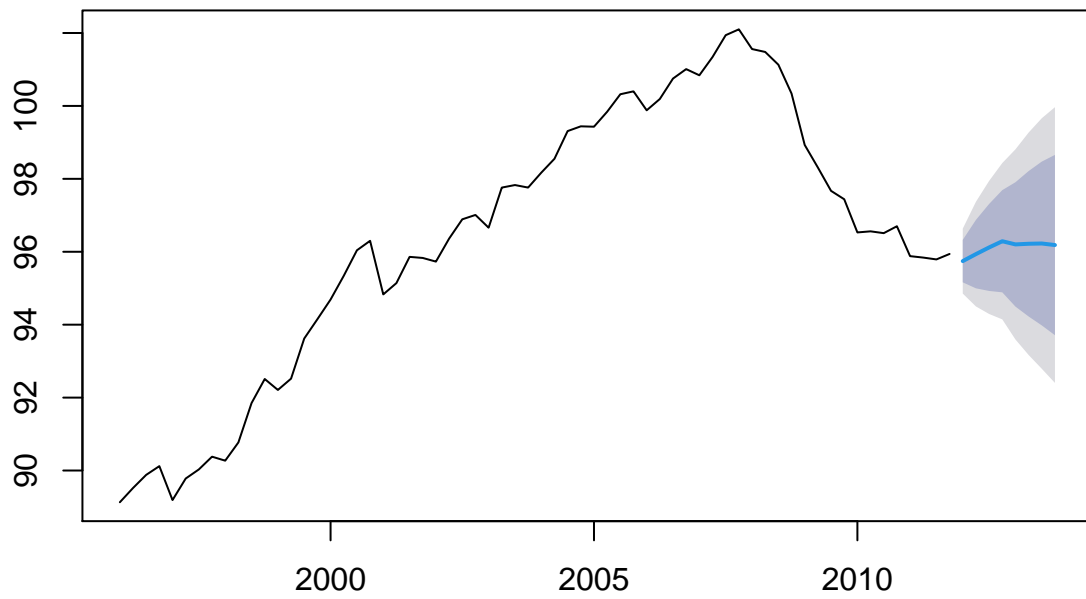
```
#Second Arima model
a2<- Arima(euretail, order=c(0,1,1), seasonal=c(0,0,2))
fit2<- fitted(a2)

plot(euretail)
lines(fit2, col=2)
```

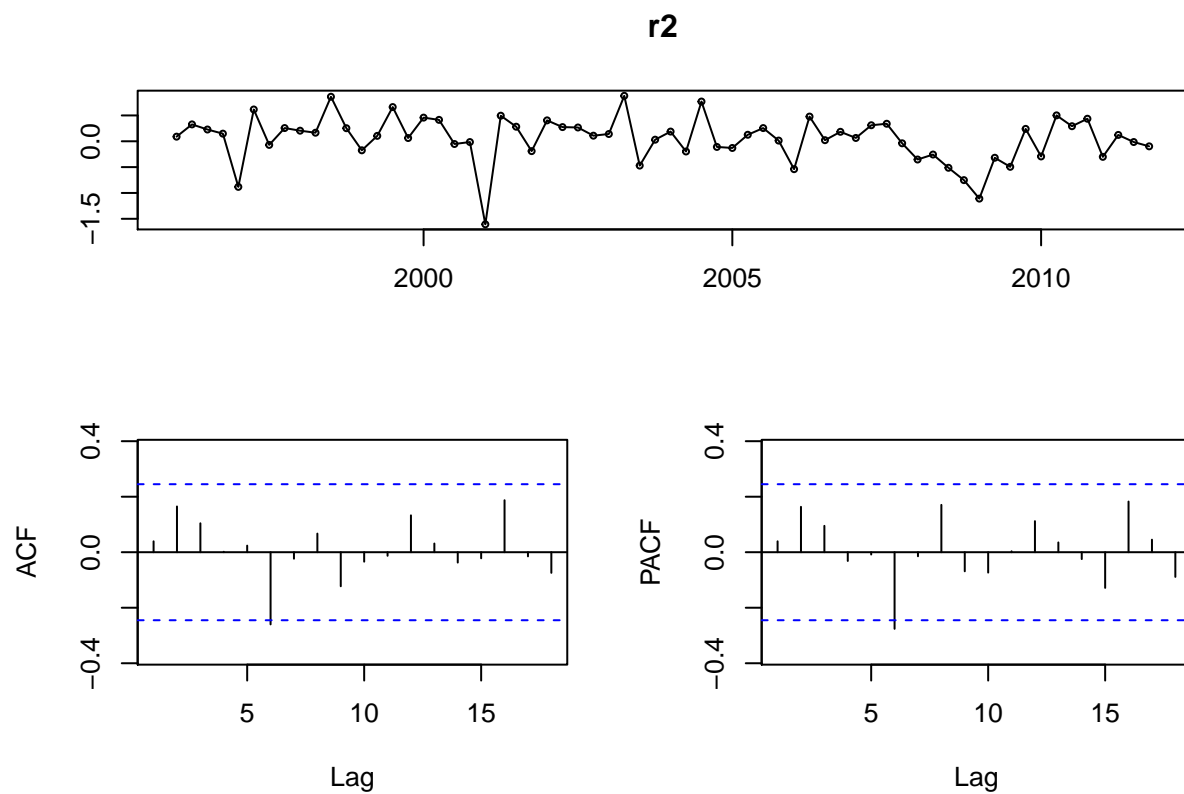


```
f2<- forecast(a2)  
plot(f2)
```

Forecasts from ARIMA(0,1,1)(0,0,2)[4]

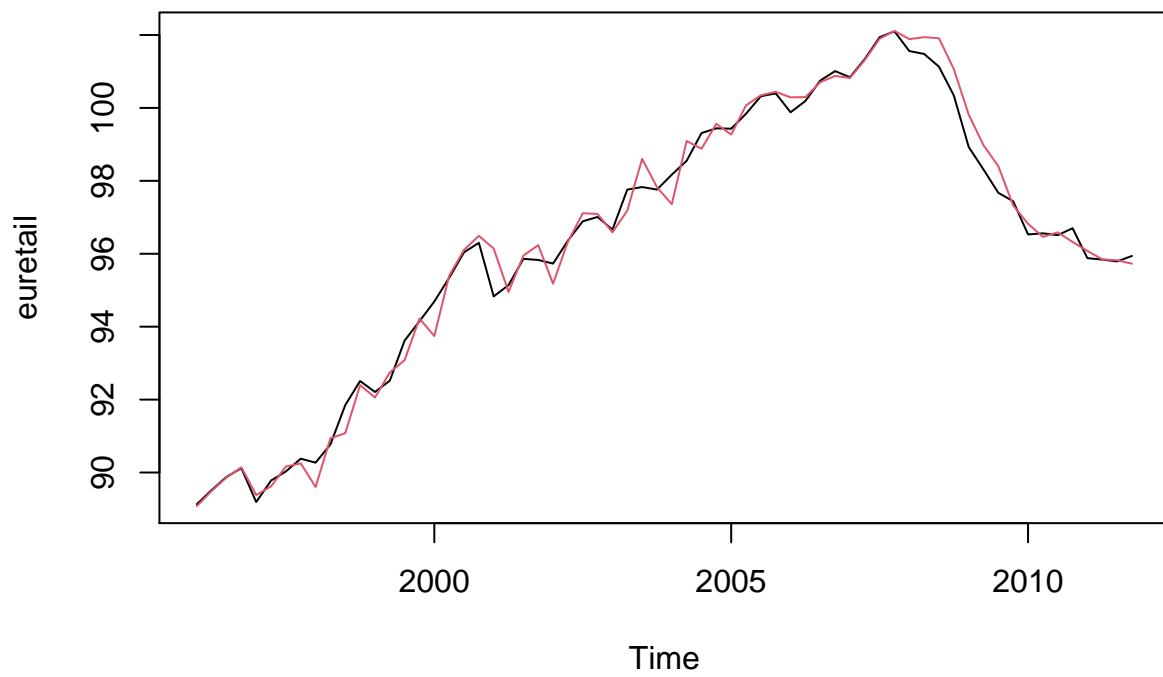


```
r2<- residuals(a2)
tsdisplay(r2)
```

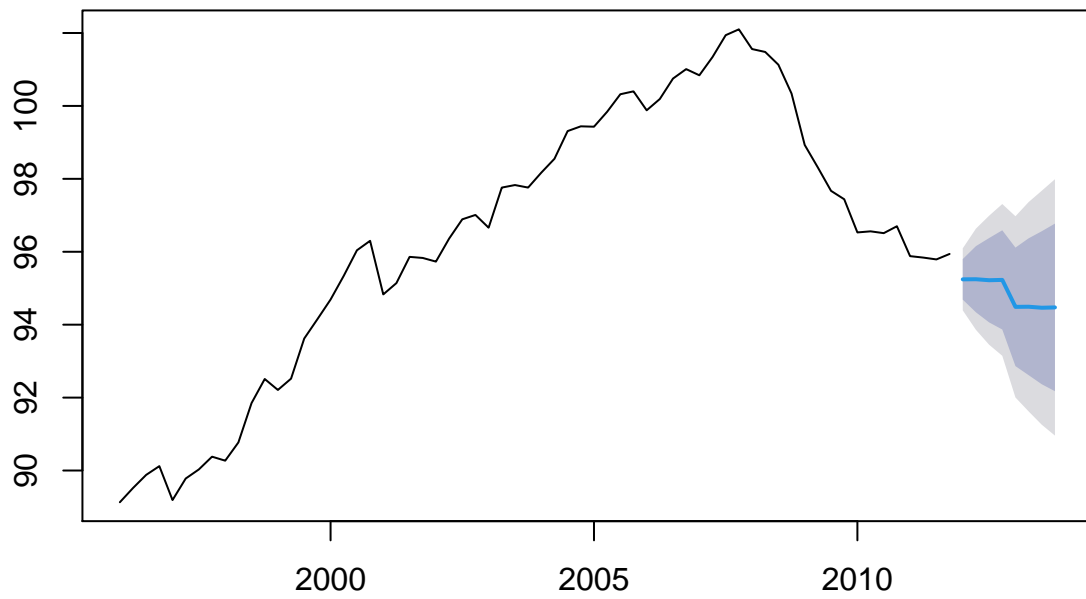
```
#Third Arima model
a3<- Arima(euretail, order=c(0,1,1), seasonal=c(0,1,1))
fit3<- fitted(a3)

plot(euretail)
lines(fit3, col=2)
```

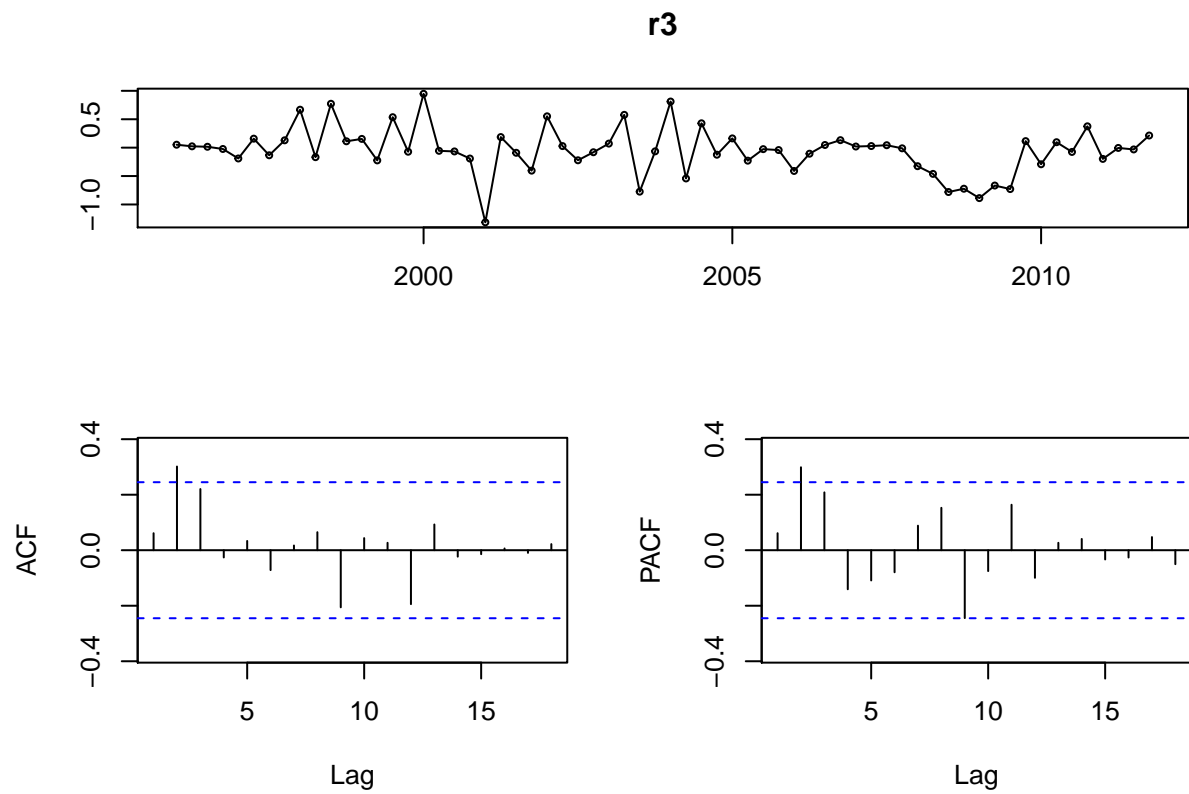


```
f3<- forecast(a3)  
plot(f3)
```

Forecasts from ARIMA(0,1,1)(0,1,1)[4]

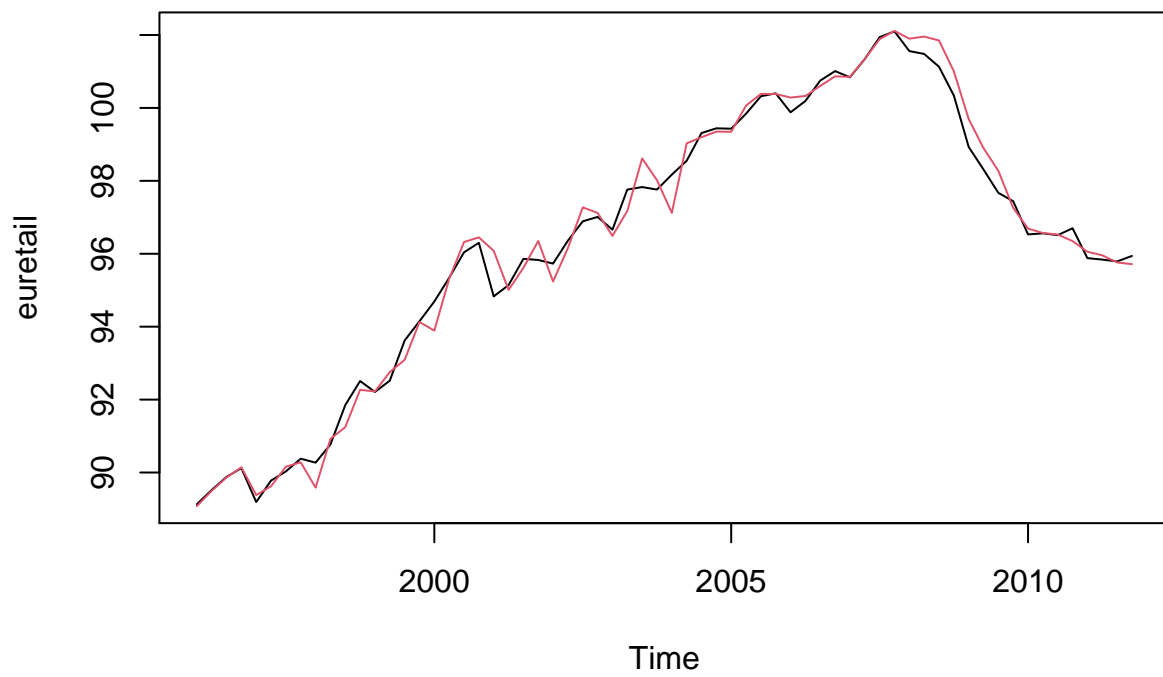


```
r3<- residuals(a3)
tsdisplay(r3)
```



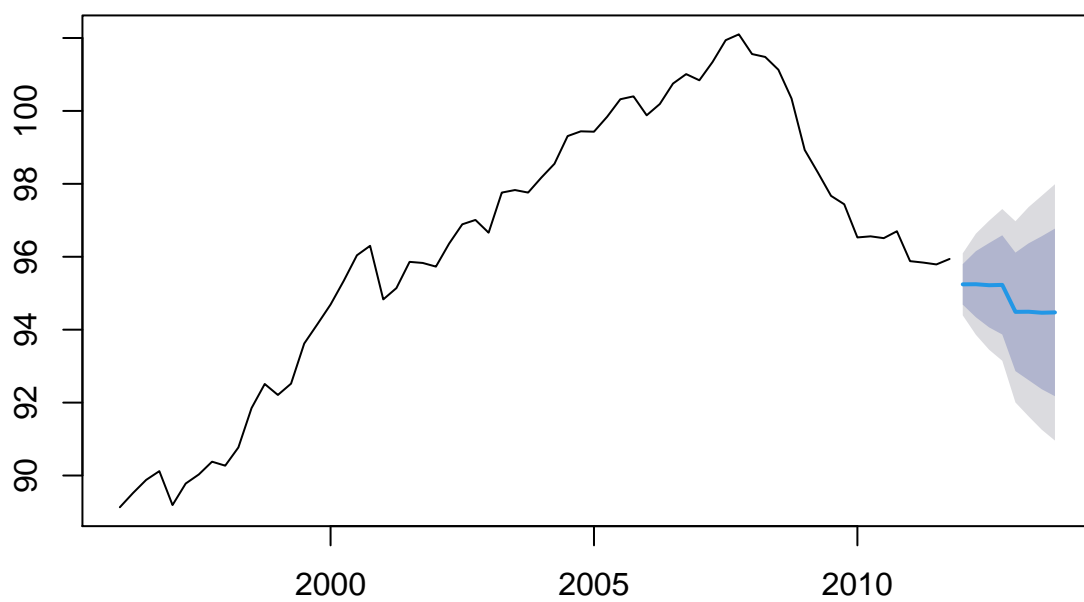
```
#Fourth Arima model
a4<- Arima(euretail, order=c(0,1,2), seasonal=c(0,1,1))
fit4<- fitted(a4)

plot(euretail)
lines(fit4, col=2)
```

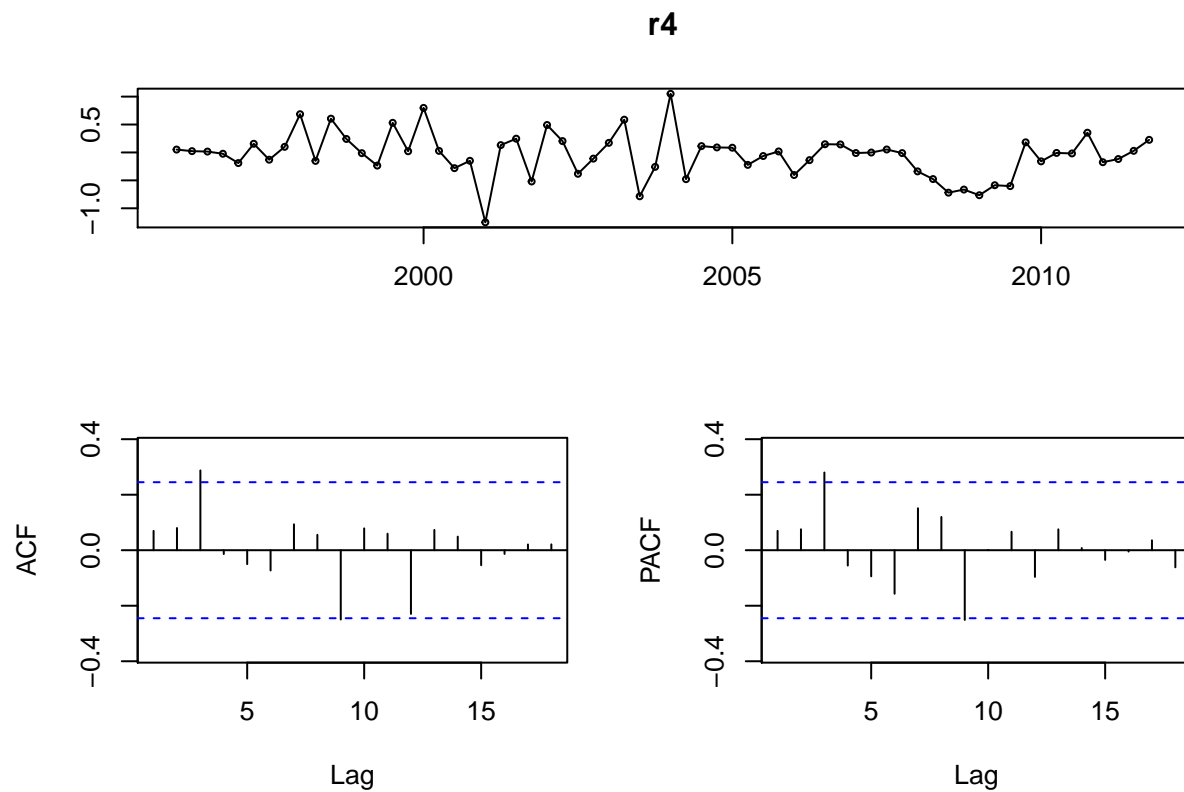


```
f4<- forecast(a4)  
plot(f3)
```

Forecasts from ARIMA(0,1,1)(0,1,1)[4]



```
r4<- residuals(a4)
tsdisplay(r4)
```

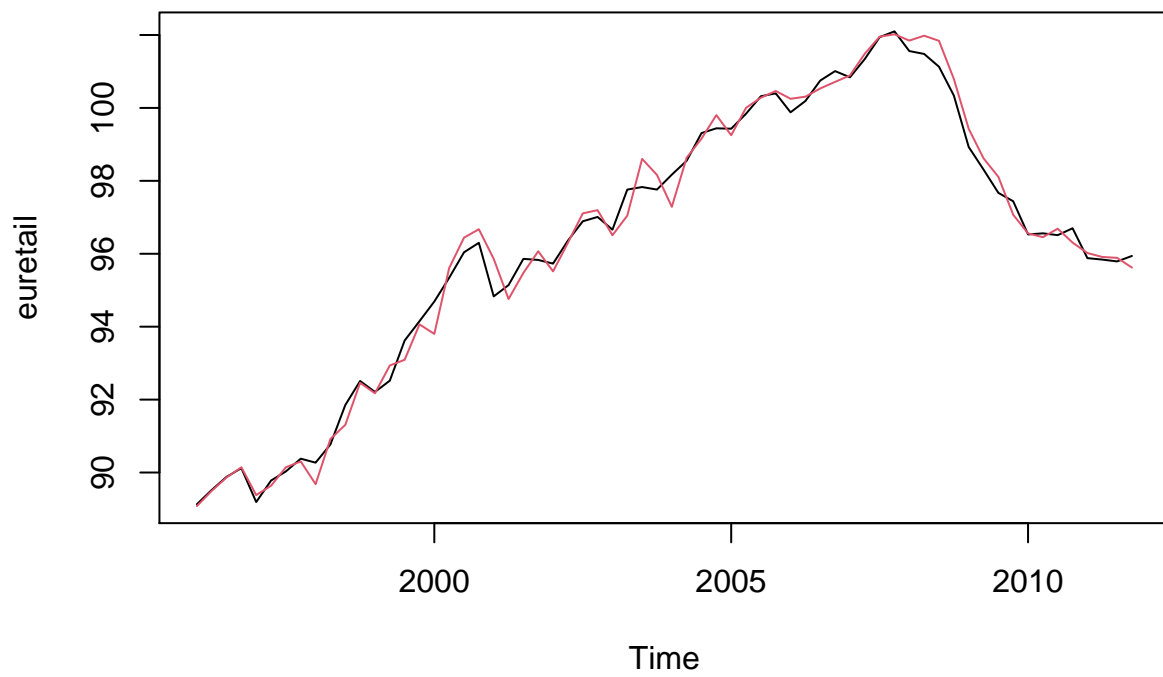


```
#Fifth Arima model (auto.arima)
auto.a<- auto.arima(euretail)
auto.a
```

```
## Series: euretail
## ARIMA(0,1,3)(0,1,1)[4]
##
## Coefficients:
##          ma1      ma2      ma3      sma1
##          0.2630  0.3694  0.4200 -0.6636
## s.e.      0.1237  0.1255  0.1294  0.1545
##
## sigma^2 estimated as 0.156:  log likelihood=-28.63
## AIC=67.26   AICc=68.39   BIC=77.65
```

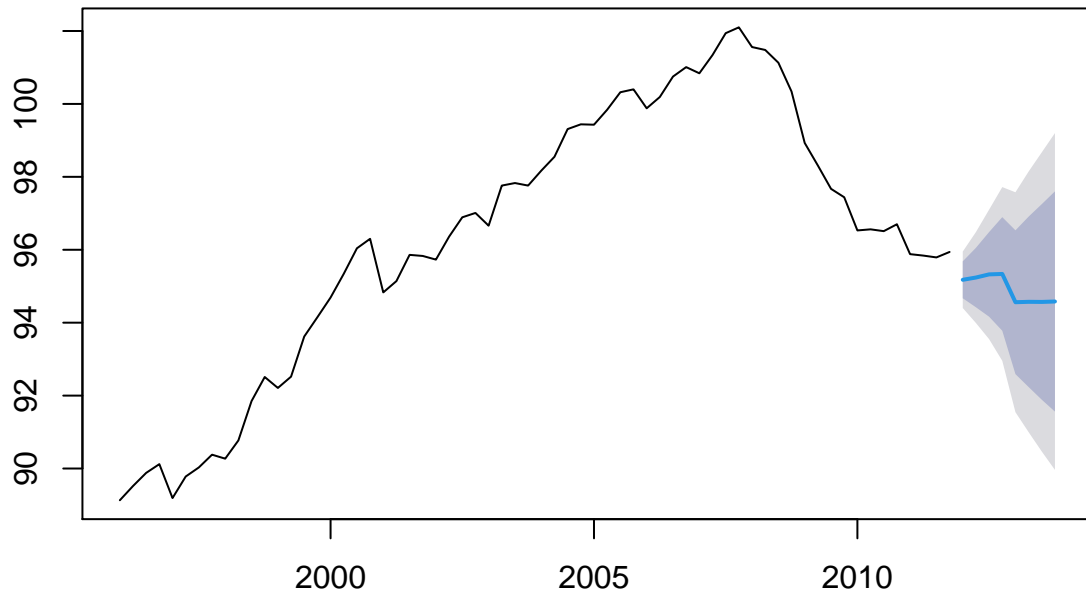
```
fit5<- fitted(auto.a)

plot(euretail)
lines(fit5, col=2)
```

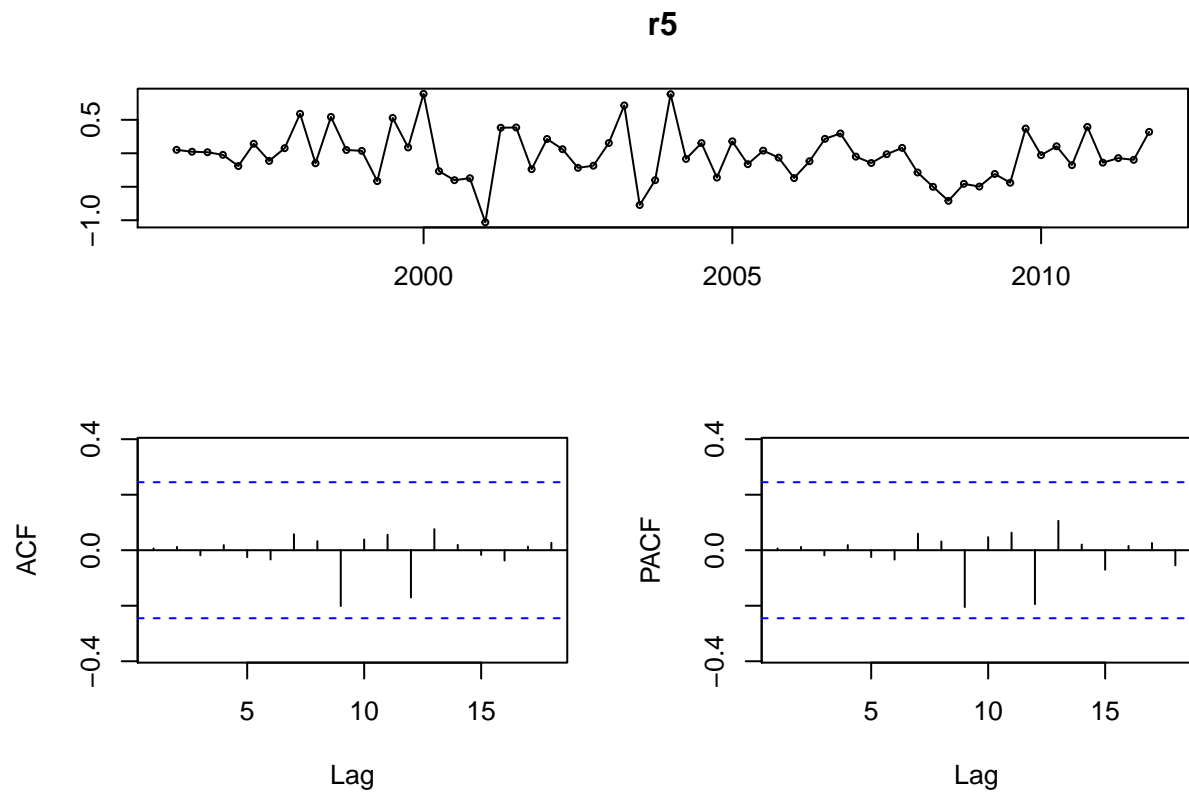


```
f5<- forecast(auto.a)  
plot(f5)
```


Forecasts from ARIMA(0,1,3)(0,1,1)[4]

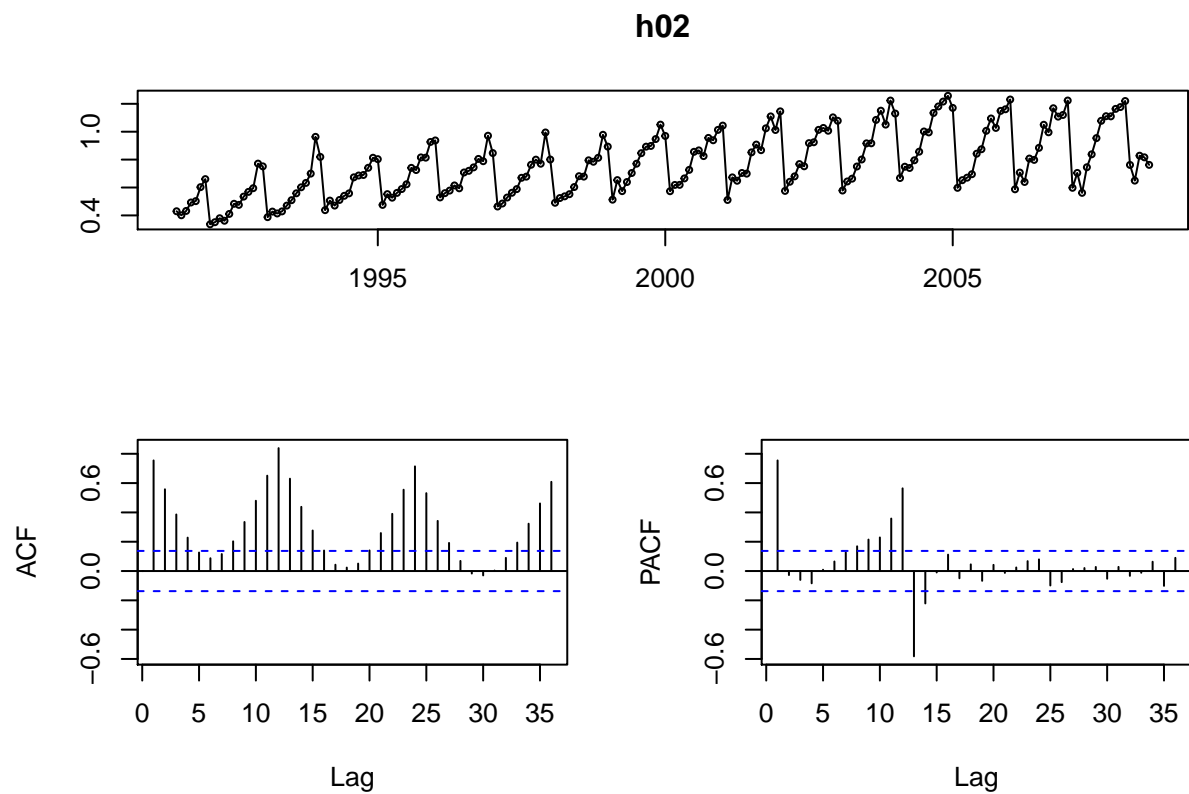


```
r5<- residuals(auto.a)  
tsdisplay(r5)
```



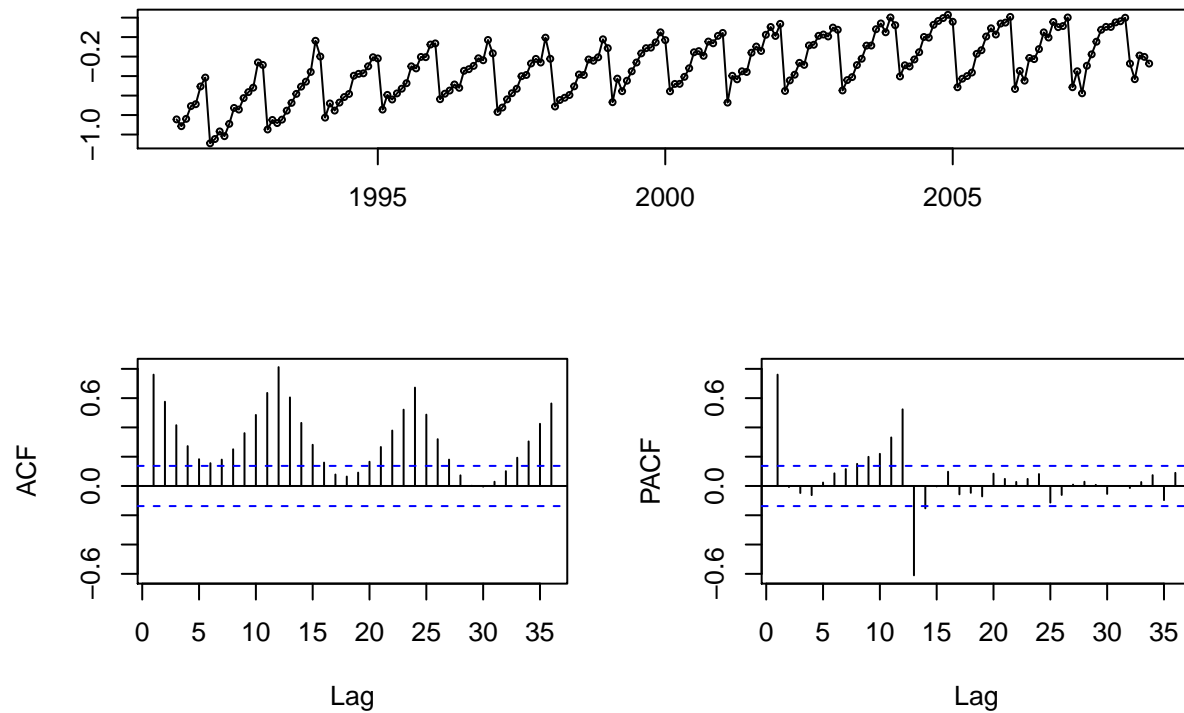
drug sales in Australia (July 1991- June 2008)

```
#Explore and transform the data
tsdisplay(h02)
```



```
#Perform log transformation to stabilize variance  
lh02<- log(h02)  
tsdisplay(lh02)
```

lh02



```
str(lh02)
```

```
## Time-Series [1:204] from 1992 to 2008: -0.844 -0.914 -0.839 -0.708 -0.688 ...
```

```
#Use of function window() to create training and test set, and also to create a portion of dataset
lh.train<- window(lh02, end=2005)
aa<- auto.arima(lh.train)
summary(aa)
```

```
## Series: lh.train
## ARIMA(2,1,0)(0,1,2)[12]
##
## Coefficients:
##          ar1          ar2          sma1          sma2
##       -0.7355   -0.3867   -0.3957   -0.2383
## s.e.    0.0781    0.0796    0.0840    0.0809
##
## sigma^2 estimated as 0.003243: log likelihood=216.11
## AIC=-422.21   AICc=-421.8   BIC=-407.16
##
## Training set error measures:
##              ME          RMSE          MAE          MPE          MAPE          MASE
## Training set -0.003016663 0.05389992 0.04025559 -8.925192 54.2031 0.4753713
##              ACF1
## Training set 0.02203246
```

```
#To see only the AIC we can use: AIC(aa)
```

```
#Fitting
```

```
fit<- fitted(aa)
```

```
fit
```

##	Jan	Feb	Mar	Apr	May
## 1991					
## 1992	-0.415593068	-1.089581381	-1.045657038	-0.967874639	-1.016419320
## 1993	-0.239736803	-0.907824435	-0.867481879	-0.820869143	-0.868560476
## 1994	-0.121298779	-0.777210818	-0.706708898	-0.742944037	-0.710759466
## 1995	-0.155745778	-0.840463988	-0.740496316	-0.690595212	-0.597418199
## 1996	-0.046598482	-0.645789473	-0.521458773	-0.555703319	-0.523572223
## 1997	-0.047853629	-0.686286300	-0.644956075	-0.685142082	-0.628755059
## 1998	-0.120659810	-0.753811915	-0.674234983	-0.640007348	-0.552141862
## 1999	-0.147602771	-0.665319849	-0.594649350	-0.524708466	-0.478744660
## 2000	0.021958925	-0.570879984	-0.413325259	-0.490911013	-0.405605672
## 2001	-0.007037603	-0.558312440	-0.512986353	-0.466552579	-0.417463871
## 2002	0.120834195	-0.541443507	-0.362726117	-0.383837526	-0.334537827
## 2003	0.107941212	-0.544794258	-0.426775247	-0.414046175	-0.311290291
## 2004	0.170840792	-0.478749008	-0.321305497	-0.302643563	-0.175683679
## 2005	0.217568293				
##	Jun	Jul	Aug	Sep	Oct
## 1991		-0.843959387	-0.913756360	-0.838847124	-0.708203455
## 1992	-0.890642280	-0.730331099	-0.781123805	-0.685107794	-0.548489338
## 1993	-0.745773189	-0.595569587	-0.621529082	-0.527161599	-0.453041247
## 1994	-0.602081575	-0.498472904	-0.461426918	-0.391475443	-0.318560710
## 1995	-0.511664413	-0.435850438	-0.332828805	-0.289888292	-0.259298834
## 1996	-0.460053501	-0.369397739	-0.255969027	-0.272480485	-0.167011261
## 1997	-0.602735191	-0.414872538	-0.361364937	-0.317744719	-0.246335184
## 1998	-0.513279294	-0.405805192	-0.354543340	-0.267855722	-0.203054255
## 1999	-0.378550441	-0.277771283	-0.225545342	-0.094688707	-0.076813893
## 2000	-0.347935170	-0.220991459	-0.127823725	-0.060290844	-0.058271171
## 2001	-0.273585560	-0.157800892	-0.143964127	-0.139382416	-0.015425707
## 2002	-0.310142914	-0.107095199	-0.033002181	-0.050719577	0.081719821
## 2003	-0.288213468	-0.094370581	-0.059321923	0.009981692	0.053497615
## 2004	-0.142124455	-0.014967499	0.004419517	0.116631936	0.193834937
## 2005					
##	Nov	Dec			
## 1991	-0.688462123	-0.506617524			
## 1992	-0.510845707	-0.323024206			
## 1993	-0.383182324	-0.144683463			
## 1994	-0.225604274	0.002997144			
## 1995	-0.177301035	-0.049012888			
## 1996	-0.184042021	-0.020307490			
## 1997	-0.226242271	-0.018161226			
## 1998	-0.225192536	0.014909849			
## 1999	-0.044201165	0.121519774			
## 2000	-0.056612767	0.075605368			
## 2001	-0.004300547	0.142648969			
## 2002	0.124673051	0.113985019			
## 2003	0.078308215	0.212635853			
## 2004	0.143149020	0.274802511			

```
## 2005
```

```
lh.test<- window(lh02, start=2005)
lh.test
```

```
##           Jan           Feb           Mar           Apr           May
## 2005  0.157593319 -0.514768386 -0.426806218 -0.399724119 -0.363486663
## 2006  0.207575800 -0.532500503 -0.346782606 -0.446848198 -0.213929878
## 2007  0.201567657 -0.514577654 -0.350411742 -0.576680567 -0.294024812
## 2008  0.198802497 -0.272042346 -0.431652525 -0.188878607 -0.203028473
##           Jun           Jul           Aug           Sep           Oct
## 2005 -0.171662962 -0.134290538  0.006475985  0.090513238  0.026683800
## 2006 -0.225684276 -0.122945337  0.048454870 -0.004300233  0.155509471
## 2007 -0.176815941 -0.046940676  0.075311060  0.105243958  0.104341222
## 2008 -0.271628949
##           Nov           Dec
## 2005  0.139093893  0.149033610
## 2006  0.102590884  0.113376006
## 2007  0.151462183  0.162619574
## 2008
```

```
#Accuracy measures in test set (?)
accuracy(fit, lh.test)
```

```
##           ME           RMSE           MAE           MPE           MAPE
## Test set -0.05997497 0.05997497 0.05997497 -38.0568 38.0568
```

```
#ME=mean(lh.test-fit)
#RMSE=sqrt(mean((lh.test-fit)**2))
#MAE=mean(abs(lh.test-fit))
```

ARMAX model (From here, was treated in class)

Data on US personal consumption and income

In ARMAX model (ARIMA+regression model) we want to extend the ARIMA by combining regression model and ARIMA model to obtain regression with ARIMA errors. In short, we add to the ARIMA some variables useful to understand better the behavior of our series. “X” stands for regression t.s.. We have to imagine a regression model with a focus on the errors using ARIMA: $y = x + e$ (y = regression part + ARIMA part)

```
uschange
```

```
##           Consumption           Income Production           Savings Unemployment
## 1970 Q1  0.61598622  0.97226104 -2.45270031  4.81031150           0.9
## 1970 Q2  0.46037569  1.16908472 -0.55152509  7.28799234           0.5
## 1970 Q3  0.87679142  1.55327055 -0.35870786  7.28901306           0.5
## 1970 Q4 -0.27424514 -0.25527238 -2.18545486  0.98522964           0.7
## 1971 Q1  1.89737076  1.98715363  1.90973412  3.65777061          -0.1
## 1971 Q2  0.91199291  1.44733417  0.90153584  6.05134180          -0.1
## 1971 Q3  0.79453885  0.53181193  0.30801942 -0.44583221           0.1
```

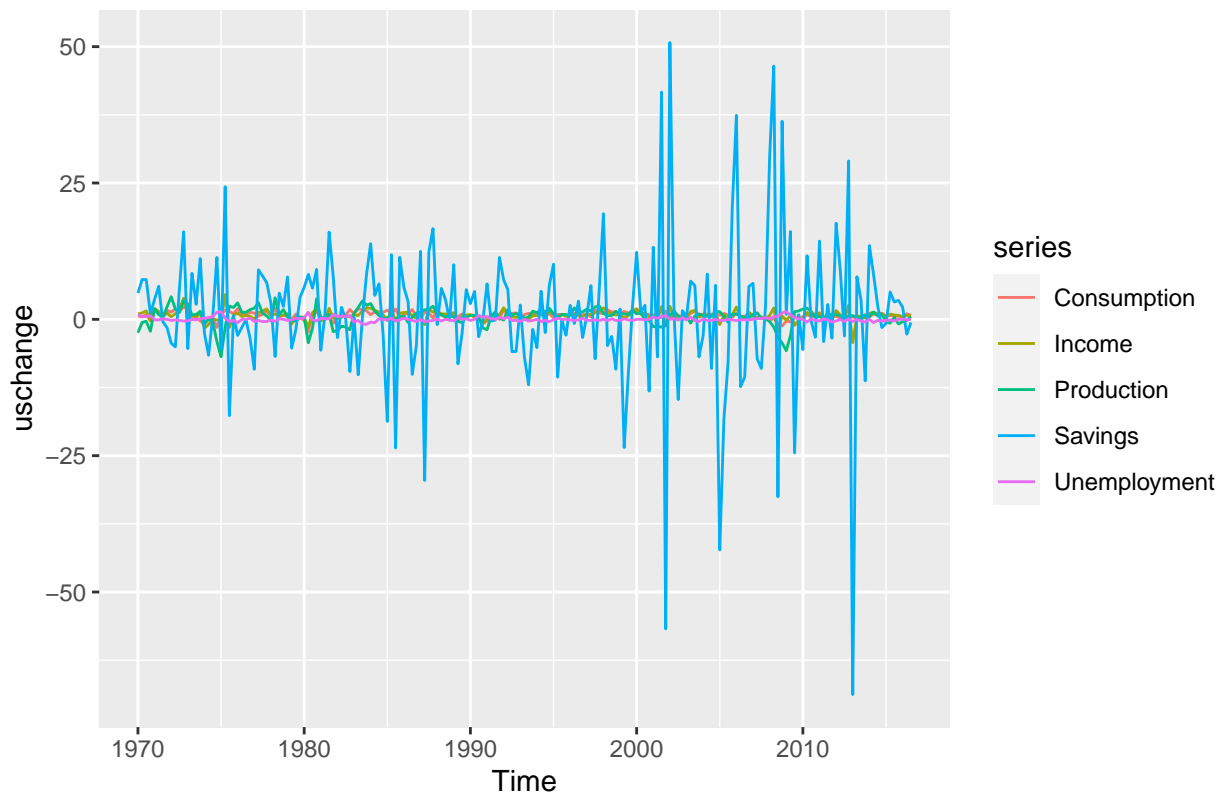
## 1971 Q4	1.64858747	1.16012514	2.29130441	-1.53087186	0.0
## 1972 Q1	1.31372218	0.45701150	4.14957387	-4.35859438	-0.2
## 1972 Q2	1.89147495	1.01662441	1.89062398	-5.05452579	-0.1
## 1972 Q3	1.53071400	1.90410126	1.27335290	5.80995904	-0.2
## 1972 Q4	2.31829471	3.89025866	3.43689207	16.04471706	-0.3
## 1973 Q1	1.81073916	0.70825266	2.79907636	-5.34886849	-0.3
## 1973 Q2	-0.04173996	0.79430954	0.81768862	8.42603436	0.0
## 1973 Q3	0.35423556	0.43381827	0.86899693	2.75879565	-0.1
## 1973 Q4	-0.29163216	1.09380979	1.47296187	11.14642986	0.1
## 1974 Q1	-0.87702794	-1.66168482	-0.88248358	-2.53351449	0.2
## 1974 Q2	0.35113555	-0.93835321	0.07427919	-6.59264464	0.3
## 1974 Q3	0.40959770	0.09448779	-0.41314971	0.51717884	0.5
## 1974 Q4	-1.47580863	-0.12259599	-4.06411893	11.34339540	1.3
## 1975 Q1	0.83225762	-0.16369546	-6.85103912	-5.47619069	1.4
## 1975 Q2	1.65583461	4.53650956	-1.33129558	24.30960536	0.2
## 1975 Q3	1.41942029	-1.46376532	2.42435972	-17.65616104	-0.4
## 1975 Q4	1.05437932	0.76166351	2.16904208	0.64809041	-0.2
## 1976 Q1	1.97998024	1.16825761	3.02720471	-2.95006644	-0.6
## 1976 Q2	0.91391607	0.51729906	1.27881101	-1.47455755	0.0
## 1976 Q3	1.05532326	0.73370026	1.30386487	-0.06754475	0.0
## 1976 Q4	1.29889825	0.59458339	1.77537765	-3.57672239	0.2
## 1977 Q1	1.13637586	-0.03108003	2.05516067	-9.16055658	-0.4
## 1977 Q2	0.54994073	1.23808955	3.05838507	9.09050404	-0.2
## 1977 Q3	0.94985262	1.51880293	1.10308888	7.94495719	-0.4
## 1977 Q4	1.49599724	1.91456240	0.63346850	6.69627648	-0.4
## 1978 Q1	0.57549599	0.70266687	-0.29339056	2.92296383	-0.1
## 1978 Q2	2.11120960	0.98314132	3.94815264	-6.81114259	-0.4
## 1978 Q3	0.41796279	0.71992620	0.87114701	4.79207162	0.1
## 1978 Q4	0.79792710	0.78553605	1.78447991	2.37118400	0.0
## 1979 Q1	0.50584598	1.05755946	0.42594327	7.77418337	-0.2
## 1979 Q2	-0.05775339	-0.86765105	-0.20491944	-5.28634896	-0.1
## 1979 Q3	0.97730010	0.47100340	-0.29723637	-1.84549644	0.2
## 1979 Q4	0.26826982	0.44037974	0.33560928	4.04959810	0.1
## 1980 Q1	-0.15391875	0.33827686	0.41056141	5.86168864	0.3
## 1980 Q2	-2.27411019	-1.46388507	-4.30076832	8.24322919	1.3
## 1980 Q3	1.07188123	1.21301507	-1.64181977	5.70775044	-0.1
## 1980 Q4	1.31644941	1.94243865	3.78045520	9.15098787	-0.3
## 1981 Q1	0.52472770	-0.26813406	0.24627687	-5.68139002	0.2
## 1981 Q2	-0.01728203	-0.02363025	0.30977573	0.88183993	0.1
## 1981 Q3	0.40165150	2.02680183	0.91707444	15.99035721	0.1
## 1981 Q4	-0.75287620	0.19560628	-2.25457797	7.80550650	0.9
## 1982 Q1	0.65938376	0.11969888	-2.07131293	-3.34243955	0.5
## 1982 Q2	0.36854173	0.57548997	-1.24766384	2.19400166	0.6
## 1982 Q3	0.76954464	0.53484410	-1.40050430	0.03499563	0.5
## 1982 Q4	1.80876006	0.44938311	-1.90375664	-9.57651468	0.7
## 1983 Q1	0.96802954	0.85588425	1.14655720	0.34595460	-0.5
## 1983 Q2	1.95946831	0.70632719	2.17942248	-10.17004699	-0.2
## 1983 Q3	1.73949442	1.49810999	3.36771897	0.21217916	-0.9
## 1983 Q4	1.56389332	2.13138911	2.58168445	8.21600068	-0.9
## 1984 Q1	0.84526442	2.02348788	2.89709545	13.86918150	-0.5
## 1984 Q2	1.41504495	1.64921136	1.53821324	4.38900229	-0.6
## 1984 Q3	0.76546608	1.36163845	0.72128740	6.51686089	0.1
## 1984 Q4	1.31380062	0.81927319	0.04115557	-2.87544931	0.0
## 1985 Q1	1.68655320	-0.23895759	0.32353159	-18.71008389	-0.1

## 1985 Q2	0.93436990	1.90677905	0.07020996	11.82871950	0.2
## 1985 Q3	1.90256675	-0.33536283	-0.14046924	-23.57393474	-0.3
## 1985 Q4	0.25656565	1.14181151	0.57978813	11.36628338	-0.1
## 1986 Q1	0.84304279	1.23951110	0.58132135	5.86126836	0.2
## 1986 Q2	1.11177390	1.31938549	-0.57641778	3.27551734	0.0
## 1986 Q3	1.79499406	0.70477150	0.37249329	-10.09044542	-0.2
## 1986 Q4	0.63768446	0.17977925	1.13734778	-4.82920131	-0.4
## 1987 Q1	0.01569397	0.81973366	1.30758228	12.46424452	0.0
## 1987 Q2	1.37731686	-0.97505791	1.75000563	-29.52866718	-0.4
## 1987 Q3	1.15225712	1.80185055	1.84366200	12.32810406	-0.3
## 1987 Q4	0.21016439	1.32743427	2.40645058	16.63076101	-0.2
## 1988 Q1	1.76316026	1.44861875	0.92013121	-0.96896505	0.0
## 1988 Q2	0.73053714	1.02084894	0.87316353	5.67776867	-0.3
## 1988 Q3	0.85083233	0.95820336	0.38103668	3.64649867	0.0
## 1988 Q4	1.13789838	0.96207024	0.70292025	-0.19730358	-0.1
## 1989 Q1	0.46064152	1.22693023	0.43372685	10.01461545	-0.3
## 1989 Q2	0.46937808	-0.29489091	-0.36675732	-8.15576525	0.3
## 1989 Q3	0.98950145	0.67822897	-0.62142121	-2.48622554	0.0
## 1989 Q4	0.43942767	0.80025832	0.42443392	5.44681102	0.1
## 1990 Q1	0.85543417	0.83939484	0.68265169	2.87544931	-0.2
## 1990 Q2	0.31230451	0.59572848	0.77446547	5.10951644	0.0
## 1990 Q3	0.40261313	0.03740765	0.41944800	-3.17767248	0.7
## 1990 Q4	-0.75910716	-0.79479735	-1.57345296	-0.17953326	0.4
## 1991 Q1	-0.34535008	0.21183290	-1.91422028	6.49315257	0.5
## 1991 Q2	0.83564224	0.69043356	0.59131506	-0.30920615	0.1
## 1991 Q3	0.48439843	0.36205181	1.36255645	-0.14086493	0.0
## 1991 Q4	-0.02626579	0.85100324	0.21710308	11.34193010	0.4
## 1992 Q1	1.85996999	2.12421067	-0.13365365	7.23265150	0.1
## 1992 Q2	0.68354371	1.04095059	1.76874773	5.46708666	0.4
## 1992 Q3	1.07661214	0.43562041	0.76167388	-5.93646090	-0.2
## 1992 Q4	1.18372396	0.34210852	1.05024577	-5.88618856	-0.2
## 1993 Q1	0.37817936	0.55877186	0.87901471	2.63464703	-0.4
## 1993 Q2	0.89392729	0.17627103	0.21755108	-6.91664675	0.0
## 1993 Q3	1.09813766	0.05868803	0.40135891	-11.99337844	-0.3
## 1993 Q4	0.88122025	0.65496353	1.49618275	-1.83708870	-0.2
## 1994 Q1	1.14064791	0.69846579	1.22213656	-5.18600629	0.0
## 1994 Q2	0.77176225	1.05367166	1.78250275	5.15609751	-0.4
## 1994 Q3	0.77214364	0.59247377	1.26718100	-2.42215898	-0.2
## 1994 Q4	1.07014805	1.38110661	2.04370404	6.32351898	-0.4
## 1995 Q1	0.26420505	0.94873528	1.02552601	10.11514398	-0.1
## 1995 Q2	0.89311141	0.22780635	0.33785685	-10.60541172	0.2
## 1995 Q3	0.91264702	0.88957006	0.90043887	-0.11570727	0.0
## 1995 Q4	0.70025425	0.57591998	0.87467273	-2.90726686	0.0
## 1996 Q1	0.92360967	0.95255663	0.69285195	2.55933958	-0.1
## 1996 Q2	1.07997887	0.95161791	2.11134752	-0.75802112	-0.2
## 1996 Q3	0.60055799	0.79369738	1.24418680	3.33843952	-0.1
## 1996 Q4	0.78298122	0.52035746	1.35396890	-3.33843952	0.2
## 1997 Q1	1.04949253	0.99858552	1.86714700	0.61269338	-0.2
## 1997 Q2	0.45219855	0.85103564	1.48763922	6.17532322	-0.2
## 1997 Q3	1.69654264	1.18352222	2.28632066	-7.22796452	-0.1
## 1997 Q4	1.18062797	1.42325742	2.48091341	5.43456565	-0.2
## 1998 Q1	1.02693626	2.10753052	1.10343775	19.35335228	0.0
## 1998 Q2	1.75069399	1.38767133	0.65122238	-4.81709478	-0.2
## 1998 Q3	1.30596977	1.01464427	0.72551955	-3.12983982	0.1

##	1998	Q4	1.45888615	0.80893032	1.44421674	-9.14923404	-0.2
##	1999	Q1	0.94821191	0.89173174	1.10341663	1.88735718	-0.2
##	1999	Q2	1.46971415	0.24722185	0.98574261	-23.49652903	0.1
##	1999	Q3	1.12921436	0.66729226	0.90279881	-9.86264835	-0.1
##	1999	Q4	1.45748895	1.46092242	1.75533234	2.35825225	-0.2
##	2000	Q1	1.51106759	1.95061335	0.99682019	12.28684080	0.0
##	2000	Q2	0.95508878	1.03174349	1.23293805	1.28001748	0.0
##	2000	Q3	0.96797647	1.16178668	-0.10225268	2.57390229	-0.1
##	2000	Q4	0.88629738	0.33725343	-0.20388383	-13.16296208	0.0
##	2001	Q1	0.42159086	0.84865826	-1.35143911	13.22491995	0.4
##	2001	Q2	0.25689982	-0.08818148	-1.25954437	-6.89043916	0.2
##	2001	Q3	0.36381084	2.33678920	-1.44101744	41.66826457	0.5
##	2001	Q4	1.51630321	-1.24443353	-1.06013675	-56.75209674	0.7
##	2002	Q1	0.29958257	2.40331419	0.70916406	50.75796205	0.0
##	2002	Q2	0.50899032	0.50559877	1.54280957	0.87861837	0.1
##	2002	Q3	0.69667241	-0.12828194	0.59478143	-14.70397426	-0.1
##	2002	Q4	0.53634306	0.47941927	-0.05776556	1.58733492	0.3
##	2003	Q1	0.43826169	0.27834026	0.53922789	0.49744834	-0.1
##	2003	Q2	1.10719086	1.43729445	-0.69876172	7.00891625	0.4
##	2003	Q3	1.46377882	1.62544947	0.60727351	6.18413150	-0.2
##	2003	Q4	0.77334046	0.40353864	1.00599126	-6.89274778	-0.4
##	2004	Q1	0.96768535	0.72653162	0.65792806	-2.96152040	0.1
##	2004	Q2	0.64760607	0.98056746	0.57461780	8.30885627	-0.2
##	2004	Q3	0.95117167	0.52450113	0.56330030	-8.99318286	-0.2
##	2004	Q4	1.02041702	1.24238706	1.38522763	6.23585017	0.0
##	2005	Q1	0.76172556	-0.96827007	1.39435718	-42.28191228	-0.2
##	2005	Q2	1.08136588	0.78835467	0.50586367	-18.27592893	-0.2
##	2005	Q3	0.77186494	0.51136949	-0.50305848	-7.87665229	0.0
##	2005	Q4	0.37591485	0.82191843	0.93365010	20.37236078	-0.1
##	2006	Q1	1.11522822	2.25904474	0.95057853	37.40653542	-0.2
##	2006	Q2	0.53100554	0.14987813	0.59636010	-12.34810568	-0.1
##	2006	Q3	0.58208747	0.28490722	0.33552773	-10.55276140	-0.1
##	2006	Q4	1.01434389	1.30059162	0.25603401	6.03100080	-0.1
##	2007	Q1	0.52486184	0.65373993	0.91794957	6.60516929	0.0
##	2007	Q2	0.33874119	0.19260870	1.19594247	-7.23648452	0.2
##	2007	Q3	0.44391875	0.26238732	0.22356909	-9.00674555	0.1
##	2007	Q4	0.12505584	0.08392938	0.16424632	2.32887238	0.3
##	2008	Q1	-0.20652548	0.71926565	-0.42872571	29.83728599	0.1
##	2008	Q2	0.16783443	2.08693775	-1.41297022	46.43989041	0.5
##	2008	Q3	-0.72499446	-2.32611860	-3.26349945	-32.53252494	0.5
##	2008	Q4	-1.21068558	0.64019534	-4.35417741	36.31240490	1.2
##	2009	Q1	-0.34354370	-0.18888849	-5.75045075	0.92306020	1.4
##	2009	Q2	-0.45174364	0.70899368	-3.00372447	16.09059408	0.8
##	2009	Q3	0.60491332	-1.10343180	1.39880419	-24.49229966	0.3
##	2009	Q4	-0.01115014	-0.13213193	1.54400617	0.84829220	0.1
##	2010	Q1	0.53481740	0.10094986	1.88006931	-5.54399051	0.0
##	2010	Q2	0.81040406	1.29229259	2.05402479	11.65612884	-0.5
##	2010	Q3	0.64501881	0.49678098	1.42683671	-0.35208609	0.1
##	2010	Q4	1.01833874	0.69495229	0.37927209	-3.27335958	-0.2
##	2011	Q1	0.50041315	1.21571502	0.50174040	14.33860193	-0.3
##	2011	Q2	0.20141978	-0.15658108	0.21878696	-4.07705131	0.1
##	2011	Q3	0.43372599	0.52891255	1.01113866	2.72250400	-0.1
##	2011	Q4	0.33593895	0.06074719	0.85151692	-3.45447712	-0.5
##	2012	Q1	0.60108995	1.62204885	0.88651817	17.62530510	-0.3

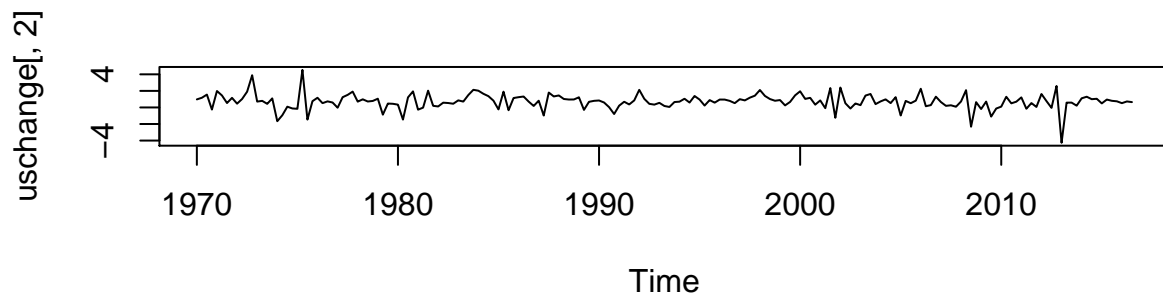
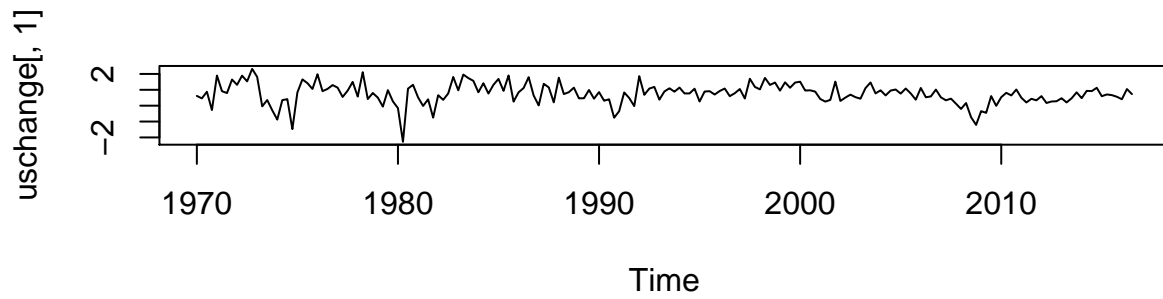
```
## 2012 Q2 0.16942956 0.76689543 0.62923586 8.96949710 0.0
## 2012 Q3 0.26416034 -0.05071452 0.07880166 -3.04922177 -0.4
## 2012 Q4 0.27877186 2.59106697 0.63305509 29.04670355 0.1
## 2013 Q1 0.46861292 -4.26525047 0.67713243 -68.78826698 -0.4
## 2013 Q2 0.20545802 0.58146541 0.30744961 7.81647729 0.0
## 2013 Q3 0.46641787 0.58328912 0.23440888 3.49400682 -0.3
## 2013 Q4 0.83917367 0.21494896 0.79208722 -11.27661450 -0.5
## 2014 Q1 0.47345118 1.10369487 0.54709166 13.52020248 0.0
## 2014 Q2 0.93375698 1.29390492 1.33801074 8.24404770 -0.6
## 2014 Q3 0.91687178 0.99853396 0.62352731 2.46195256 -0.2
## 2014 Q4 1.12533250 1.04641801 0.90355427 -1.51305022 -0.3
## 2015 Q1 0.59624005 0.49040680 -0.46710878 -0.75840017 -0.2
## 2015 Q2 0.70814389 0.95495949 -0.69702162 5.02391773 -0.1
## 2015 Q3 0.66496956 0.80166267 0.38060610 3.18092976 -0.3
## 2015 Q4 0.56167978 0.74006260 -0.84554638 3.48278601 0.0
## 2016 Q1 0.40468216 0.51902540 -0.41793048 2.23653405 0.0
## 2016 Q2 1.04770741 0.72372078 -0.20331883 -2.72150106 -0.1
## 2016 Q3 0.72959779 0.64470081 0.47491844 -0.57285793 0.0
```

```
autoplot(uschange)
```

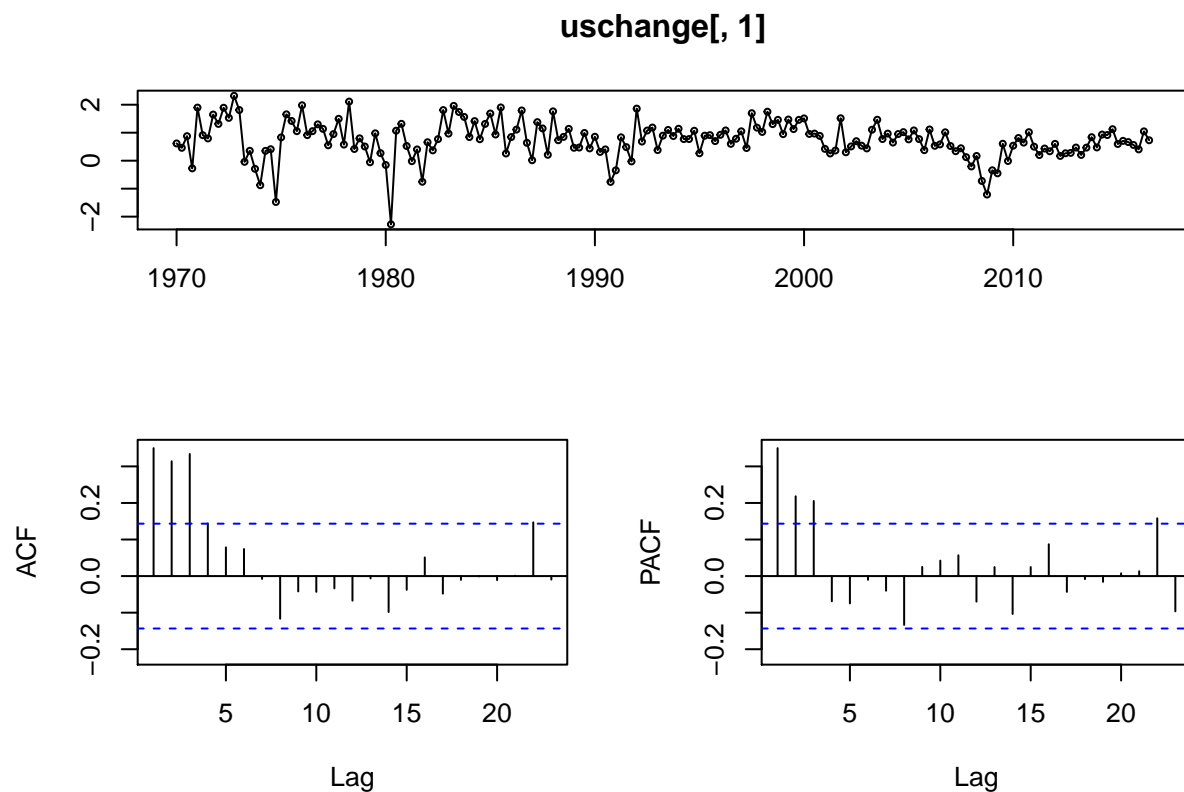


```
#Is the consumptio related to income? We expect a relation.
#Plots
par(mfrow=c(2,1))
#Consumption
```

```
plot(uschange[,1])  
#Income  
plot(uschange[,2])
```



```
par(mfrow=c(1,1))  
#More info on consumption variable  
tsdisplay(uschange[,1])
```



uschange

##		Consumption	Income	Production	Savings	Unemployment
##	1970 Q1	0.61598622	0.97226104	-2.45270031	4.81031150	0.9
##	1970 Q2	0.46037569	1.16908472	-0.55152509	7.28799234	0.5
##	1970 Q3	0.87679142	1.55327055	-0.35870786	7.28901306	0.5
##	1970 Q4	-0.27424514	-0.25527238	-2.18545486	0.98522964	0.7
##	1971 Q1	1.89737076	1.98715363	1.90973412	3.65777061	-0.1
##	1971 Q2	0.91199291	1.44733417	0.90153584	6.05134180	-0.1
##	1971 Q3	0.79453885	0.53181193	0.30801942	-0.44583221	0.1
##	1971 Q4	1.64858747	1.16012514	2.29130441	-1.53087186	0.0
##	1972 Q1	1.31372218	0.45701150	4.14957387	-4.35859438	-0.2
##	1972 Q2	1.89147495	1.01662441	1.89062398	-5.05452579	-0.1
##	1972 Q3	1.53071400	1.90410126	1.27335290	5.80995904	-0.2
##	1972 Q4	2.31829471	3.89025866	3.43689207	16.04471706	-0.3
##	1973 Q1	1.81073916	0.70825266	2.79907636	-5.34886849	-0.3
##	1973 Q2	-0.04173996	0.79430954	0.81768862	8.42603436	0.0
##	1973 Q3	0.35423556	0.43381827	0.86899693	2.75879565	-0.1
##	1973 Q4	-0.29163216	1.09380979	1.47296187	11.14642986	0.1
##	1974 Q1	-0.87702794	-1.66168482	-0.88248358	-2.53351449	0.2
##	1974 Q2	0.35113555	-0.93835321	0.07427919	-6.59264464	0.3
##	1974 Q3	0.40959770	0.09448779	-0.41314971	0.51717884	0.5
##	1974 Q4	-1.47580863	-0.12259599	-4.06411893	11.34339540	1.3
##	1975 Q1	0.83225762	-0.16369546	-6.85103912	-5.47619069	1.4
##	1975 Q2	1.65583461	4.53650956	-1.33129558	24.30960536	0.2
##	1975 Q3	1.41942029	-1.46376532	2.42435972	-17.65616104	-0.4

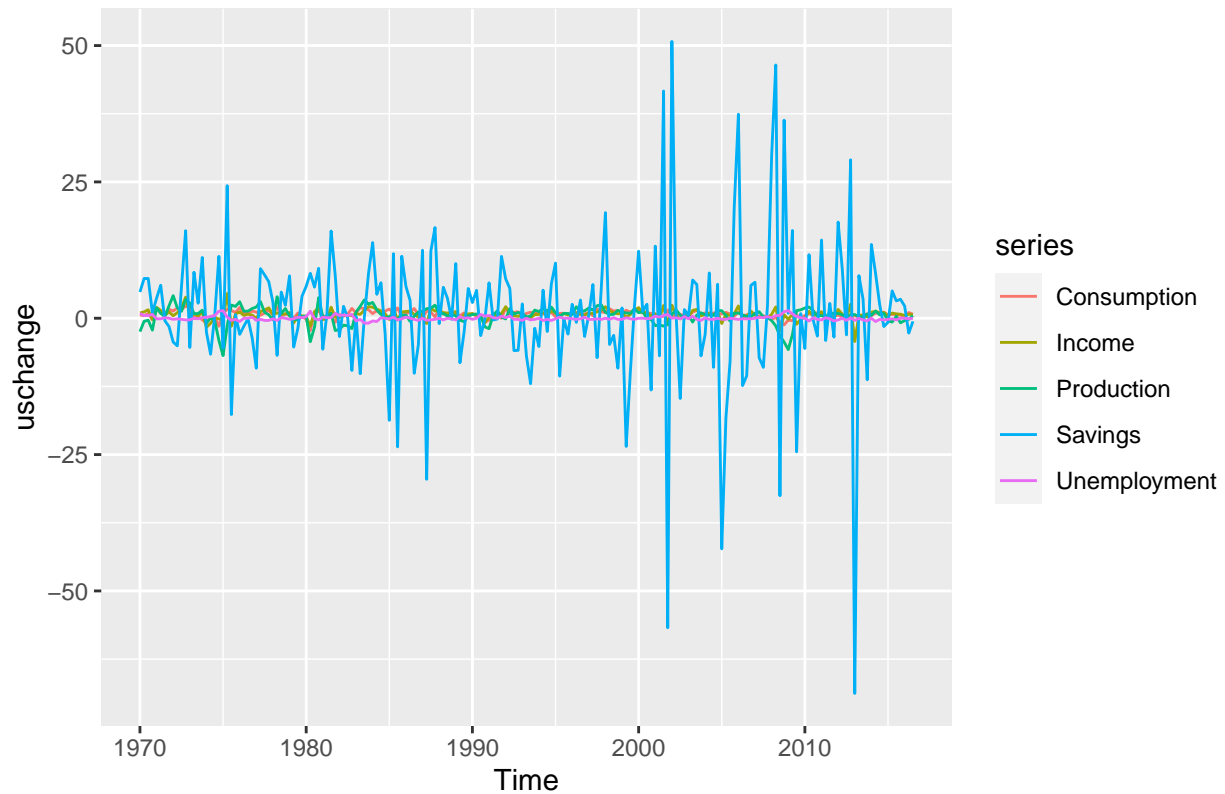
## 1975 Q4	1.05437932	0.76166351	2.16904208	0.64809041	-0.2
## 1976 Q1	1.97998024	1.16825761	3.02720471	-2.95006644	-0.6
## 1976 Q2	0.91391607	0.51729906	1.27881101	-1.47455755	0.0
## 1976 Q3	1.05532326	0.73370026	1.30386487	-0.06754475	0.0
## 1976 Q4	1.29889825	0.59458339	1.77537765	-3.57672239	0.2
## 1977 Q1	1.13637586	-0.03108003	2.05516067	-9.16055658	-0.4
## 1977 Q2	0.54994073	1.23808955	3.05838507	9.09050404	-0.2
## 1977 Q3	0.94985262	1.51880293	1.10308888	7.94495719	-0.4
## 1977 Q4	1.49599724	1.91456240	0.63346850	6.69627648	-0.4
## 1978 Q1	0.57549599	0.70266687	-0.29339056	2.92296383	-0.1
## 1978 Q2	2.11120960	0.98314132	3.94815264	-6.81114259	-0.4
## 1978 Q3	0.41796279	0.71992620	0.87114701	4.79207162	0.1
## 1978 Q4	0.79792710	0.78553605	1.78447991	2.37118400	0.0
## 1979 Q1	0.50584598	1.05755946	0.42594327	7.77418337	-0.2
## 1979 Q2	-0.05775339	-0.86765105	-0.20491944	-5.28634896	-0.1
## 1979 Q3	0.97730010	0.47100340	-0.29723637	-1.84549644	0.2
## 1979 Q4	0.26826982	0.44037974	0.33560928	4.04959810	0.1
## 1980 Q1	-0.15391875	0.33827686	0.41056141	5.86168864	0.3
## 1980 Q2	-2.27411019	-1.46388507	-4.30076832	8.24322919	1.3
## 1980 Q3	1.07188123	1.21301507	-1.64181977	5.70775044	-0.1
## 1980 Q4	1.31644941	1.94243865	3.78045520	9.15098787	-0.3
## 1981 Q1	0.52472770	-0.26813406	0.24627687	-5.68139002	0.2
## 1981 Q2	-0.01728203	-0.02363025	0.30977573	0.88183993	0.1
## 1981 Q3	0.40165150	2.02680183	0.91707444	15.99035721	0.1
## 1981 Q4	-0.75287620	0.19560628	-2.25457797	7.80550650	0.9
## 1982 Q1	0.65938376	0.11969888	-2.07131293	-3.34243955	0.5
## 1982 Q2	0.36854173	0.57548997	-1.24766384	2.19400166	0.6
## 1982 Q3	0.76954464	0.53484410	-1.40050430	0.03499563	0.5
## 1982 Q4	1.80876006	0.44938311	-1.90375664	-9.57651468	0.7
## 1983 Q1	0.96802954	0.85588425	1.14655720	0.34595460	-0.5
## 1983 Q2	1.95946831	0.70632719	2.17942248	-10.17004699	-0.2
## 1983 Q3	1.73949442	1.49810999	3.36771897	0.21217916	-0.9
## 1983 Q4	1.56389332	2.13138911	2.58168445	8.21600068	-0.9
## 1984 Q1	0.84526442	2.02348788	2.89709545	13.86918150	-0.5
## 1984 Q2	1.41504495	1.64921136	1.53821324	4.38900229	-0.6
## 1984 Q3	0.76546608	1.36163845	0.72128740	6.51686089	0.1
## 1984 Q4	1.31380062	0.81927319	0.04115557	-2.87544931	0.0
## 1985 Q1	1.68655320	-0.23895759	0.32353159	-18.71008389	-0.1
## 1985 Q2	0.93436990	1.90677905	0.07020996	11.82871950	0.2
## 1985 Q3	1.90256675	-0.33536283	-0.14046924	-23.57393474	-0.3
## 1985 Q4	0.25656565	1.14181151	0.57978813	11.36628338	-0.1
## 1986 Q1	0.84304279	1.23951110	0.58132135	5.86126836	0.2
## 1986 Q2	1.11177390	1.31938549	-0.57641778	3.27551734	0.0
## 1986 Q3	1.79499406	0.70477150	0.37249329	-10.09044542	-0.2
## 1986 Q4	0.63768446	0.17977925	1.13734778	-4.82920131	-0.4
## 1987 Q1	0.01569397	0.81973366	1.30758228	12.46424452	0.0
## 1987 Q2	1.37731686	-0.97505791	1.75000563	-29.52866718	-0.4
## 1987 Q3	1.15225712	1.80185055	1.84366200	12.32810406	-0.3
## 1987 Q4	0.21016439	1.32743427	2.40645058	16.63076101	-0.2
## 1988 Q1	1.76316026	1.44861875	0.92013121	-0.96896505	0.0
## 1988 Q2	0.73053714	1.02084894	0.87316353	5.67776867	-0.3
## 1988 Q3	0.85083233	0.95820336	0.38103668	3.64649867	0.0
## 1988 Q4	1.13789838	0.96207024	0.70292025	-0.19730358	-0.1
## 1989 Q1	0.46064152	1.22693023	0.43372685	10.01461545	-0.3

## 1989 Q2	0.46937808	-0.29489091	-0.36675732	-8.15576525	0.3
## 1989 Q3	0.98950145	0.67822897	-0.62142121	-2.48622554	0.0
## 1989 Q4	0.43942767	0.80025832	0.42443392	5.44681102	0.1
## 1990 Q1	0.85543417	0.83939484	0.68265169	2.87544931	-0.2
## 1990 Q2	0.31230451	0.59572848	0.77446547	5.10951644	0.0
## 1990 Q3	0.40261313	0.03740765	0.41944800	-3.17767248	0.7
## 1990 Q4	-0.75910716	-0.79479735	-1.57345296	-0.17953326	0.4
## 1991 Q1	-0.34535008	0.21183290	-1.91422028	6.49315257	0.5
## 1991 Q2	0.83564224	0.69043356	0.59131506	-0.30920615	0.1
## 1991 Q3	0.48439843	0.36205181	1.36255645	-0.14086493	0.0
## 1991 Q4	-0.02626579	0.85100324	0.21710308	11.34193010	0.4
## 1992 Q1	1.85996999	2.12421067	-0.13365365	7.23265150	0.1
## 1992 Q2	0.68354371	1.04095059	1.76874773	5.46708666	0.4
## 1992 Q3	1.07661214	0.43562041	0.76167388	-5.93646090	-0.2
## 1992 Q4	1.18372396	0.34210852	1.05024577	-5.88618856	-0.2
## 1993 Q1	0.37817936	0.55877186	0.87901471	2.63464703	-0.4
## 1993 Q2	0.89392729	0.17627103	0.21755108	-6.91664675	0.0
## 1993 Q3	1.09813766	0.05868803	0.40135891	-11.99337844	-0.3
## 1993 Q4	0.88122025	0.65496353	1.49618275	-1.83708870	-0.2
## 1994 Q1	1.14064791	0.69846579	1.22213656	-5.18600629	0.0
## 1994 Q2	0.77176225	1.05367166	1.78250275	5.15609751	-0.4
## 1994 Q3	0.77214364	0.59247377	1.26718100	-2.42215898	-0.2
## 1994 Q4	1.07014805	1.38110661	2.04370404	6.32351898	-0.4
## 1995 Q1	0.26420505	0.94873528	1.02552601	10.11514398	-0.1
## 1995 Q2	0.89311141	0.22780635	0.33785685	-10.60541172	0.2
## 1995 Q3	0.91264702	0.88957006	0.90043887	-0.11570727	0.0
## 1995 Q4	0.70025425	0.57591998	0.87467273	-2.90726686	0.0
## 1996 Q1	0.92360967	0.95255663	0.69285195	2.55933958	-0.1
## 1996 Q2	1.07997887	0.95161791	2.11134752	-0.75802112	-0.2
## 1996 Q3	0.60055799	0.79369738	1.24418680	3.33843952	-0.1
## 1996 Q4	0.78298122	0.52035746	1.35396890	-3.33843952	0.2
## 1997 Q1	1.04949253	0.99858552	1.86714700	0.61269338	-0.2
## 1997 Q2	0.45219855	0.85103564	1.48763922	6.17532322	-0.2
## 1997 Q3	1.69654264	1.18352222	2.28632066	-7.22796452	-0.1
## 1997 Q4	1.18062797	1.42325742	2.48091341	5.43456565	-0.2
## 1998 Q1	1.02693626	2.10753052	1.10343775	19.35335228	0.0
## 1998 Q2	1.75069399	1.38767133	0.65122238	-4.81709478	-0.2
## 1998 Q3	1.30596977	1.01464427	0.72551955	-3.12983982	0.1
## 1998 Q4	1.45888615	0.80893032	1.44421674	-9.14923404	-0.2
## 1999 Q1	0.94821191	0.89173174	1.10341663	1.88735718	-0.2
## 1999 Q2	1.46971415	0.24722185	0.98574261	-23.49652903	0.1
## 1999 Q3	1.12921436	0.66729226	0.90279881	-9.86264835	-0.1
## 1999 Q4	1.45748895	1.46092242	1.75533234	2.35825225	-0.2
## 2000 Q1	1.51106759	1.95061335	0.99682019	12.28684080	0.0
## 2000 Q2	0.95508878	1.03174349	1.23293805	1.28001748	0.0
## 2000 Q3	0.96797647	1.16178668	-0.10225268	2.57390229	-0.1
## 2000 Q4	0.88629738	0.33725343	-0.20388383	-13.16296208	0.0
## 2001 Q1	0.42159086	0.84865826	-1.35143911	13.22491995	0.4
## 2001 Q2	0.25689982	-0.08818148	-1.25954437	-6.89043916	0.2
## 2001 Q3	0.36381084	2.33678920	-1.44101744	41.66826457	0.5
## 2001 Q4	1.51630321	-1.24443353	-1.06013675	-56.75209674	0.7
## 2002 Q1	0.29958257	2.40331419	0.70916406	50.75796205	0.0
## 2002 Q2	0.50899032	0.50559877	1.54280957	0.87861837	0.1
## 2002 Q3	0.69667241	-0.12828194	0.59478143	-14.70397426	-0.1

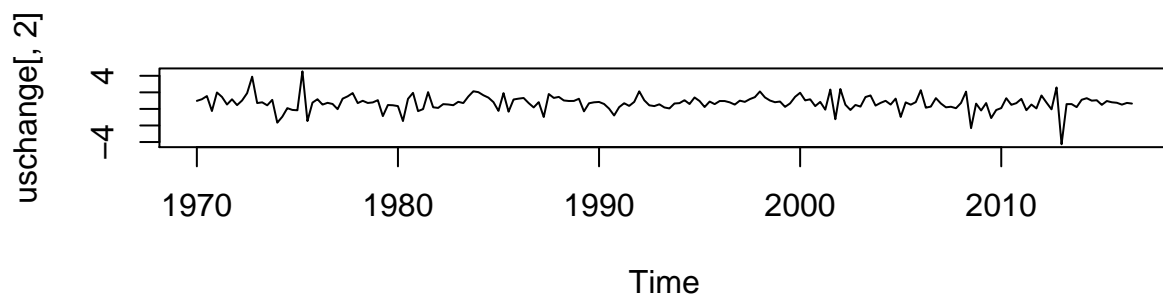
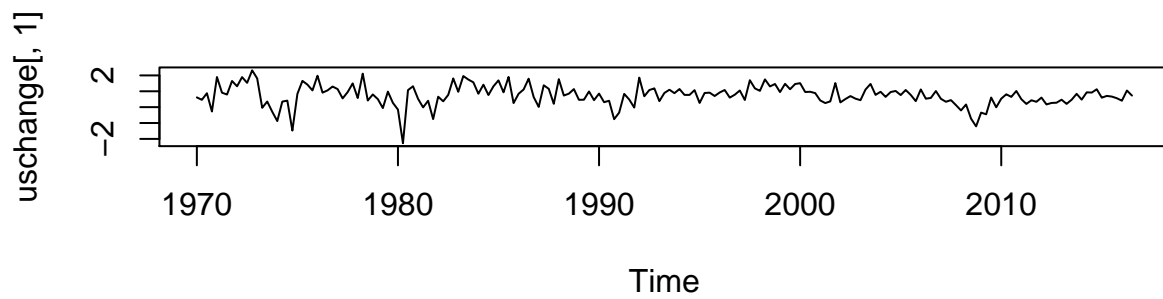
##	2002	Q4	0.53634306	0.47941927	-0.05776556	1.58733492	0.3
##	2003	Q1	0.43826169	0.27834026	0.53922789	0.49744834	-0.1
##	2003	Q2	1.10719086	1.43729445	-0.69876172	7.00891625	0.4
##	2003	Q3	1.46377882	1.62544947	0.60727351	6.18413150	-0.2
##	2003	Q4	0.77334046	0.40353864	1.00599126	-6.89274778	-0.4
##	2004	Q1	0.96768535	0.72653162	0.65792806	-2.96152040	0.1
##	2004	Q2	0.64760607	0.98056746	0.57461780	8.30885627	-0.2
##	2004	Q3	0.95117167	0.52450113	0.56330030	-8.99318286	-0.2
##	2004	Q4	1.02041702	1.24238706	1.38522763	6.23585017	0.0
##	2005	Q1	0.76172556	-0.96827007	1.39435718	-42.28191228	-0.2
##	2005	Q2	1.08136588	0.78835467	0.50586367	-18.27592893	-0.2
##	2005	Q3	0.77186494	0.51136949	-0.50305848	-7.87665229	0.0
##	2005	Q4	0.37591485	0.82191843	0.93365010	20.37236078	-0.1
##	2006	Q1	1.11522822	2.25904474	0.95057853	37.40653542	-0.2
##	2006	Q2	0.53100554	0.14987813	0.59636010	-12.34810568	-0.1
##	2006	Q3	0.58208747	0.28490722	0.33552773	-10.55276140	-0.1
##	2006	Q4	1.01434389	1.30059162	0.25603401	6.03100080	-0.1
##	2007	Q1	0.52486184	0.65373993	0.91794957	6.60516929	0.0
##	2007	Q2	0.33874119	0.19260870	1.19594247	-7.23648452	0.2
##	2007	Q3	0.44391875	0.26238732	0.22356909	-9.00674555	0.1
##	2007	Q4	0.12505584	0.08392938	0.16424632	2.32887238	0.3
##	2008	Q1	-0.20652548	0.71926565	-0.42872571	29.83728599	0.1
##	2008	Q2	0.16783443	2.08693775	-1.41297022	46.43989041	0.5
##	2008	Q3	-0.72499446	-2.32611860	-3.26349945	-32.53252494	0.5
##	2008	Q4	-1.21068558	0.64019534	-4.35417741	36.31240490	1.2
##	2009	Q1	-0.34354370	-0.18888849	-5.75045075	0.92306020	1.4
##	2009	Q2	-0.45174364	0.70899368	-3.00372447	16.09059408	0.8
##	2009	Q3	0.60491332	-1.10343180	1.39880419	-24.49229966	0.3
##	2009	Q4	-0.01115014	-0.13213193	1.54400617	0.84829220	0.1
##	2010	Q1	0.53481740	0.10094986	1.88006931	-5.54399051	0.0
##	2010	Q2	0.81040406	1.29229259	2.05402479	11.65612884	-0.5
##	2010	Q3	0.64501881	0.49678098	1.42683671	-0.35208609	0.1
##	2010	Q4	1.01833874	0.69495229	0.37927209	-3.27335958	-0.2
##	2011	Q1	0.50041315	1.21571502	0.50174040	14.33860193	-0.3
##	2011	Q2	0.20141978	-0.15658108	0.21878696	-4.07705131	0.1
##	2011	Q3	0.43372599	0.52891255	1.01113866	2.72250400	-0.1
##	2011	Q4	0.33593895	0.06074719	0.85151692	-3.45447712	-0.5
##	2012	Q1	0.60108995	1.62204885	0.88651817	17.62530510	-0.3
##	2012	Q2	0.16942956	0.76689543	0.62923586	8.96949710	0.0
##	2012	Q3	0.26416034	-0.05071452	0.07880166	-3.04922177	-0.4
##	2012	Q4	0.27877186	2.59106697	0.63305509	29.04670355	0.1
##	2013	Q1	0.46861292	-4.26525047	0.67713243	-68.78826698	-0.4
##	2013	Q2	0.20545802	0.58146541	0.30744961	7.81647729	0.0
##	2013	Q3	0.46641787	0.58328912	0.23440888	3.49400682	-0.3
##	2013	Q4	0.83917367	0.21494896	0.79208722	-11.27661450	-0.5
##	2014	Q1	0.47345118	1.10369487	0.54709166	13.52020248	0.0
##	2014	Q2	0.93375698	1.29390492	1.33801074	8.24404770	-0.6
##	2014	Q3	0.91687178	0.99853396	0.62352731	2.46195256	-0.2
##	2014	Q4	1.12533250	1.04641801	0.90355427	-1.51305022	-0.3
##	2015	Q1	0.59624005	0.49040680	-0.46710878	-0.75840017	-0.2
##	2015	Q2	0.70814389	0.95495949	-0.69702162	5.02391773	-0.1
##	2015	Q3	0.66496956	0.80166267	0.38060610	3.18092976	-0.3
##	2015	Q4	0.56167978	0.74006260	-0.84554638	3.48278601	0.0
##	2016	Q1	0.40468216	0.51902540	-0.41793048	2.23653405	0.0

```
## 2016 Q2  1.04770741  0.72372078 -0.20331883 -2.72150106      -0.1
## 2016 Q3  0.72959779  0.64470081  0.47491844 -0.57285793       0.0
```

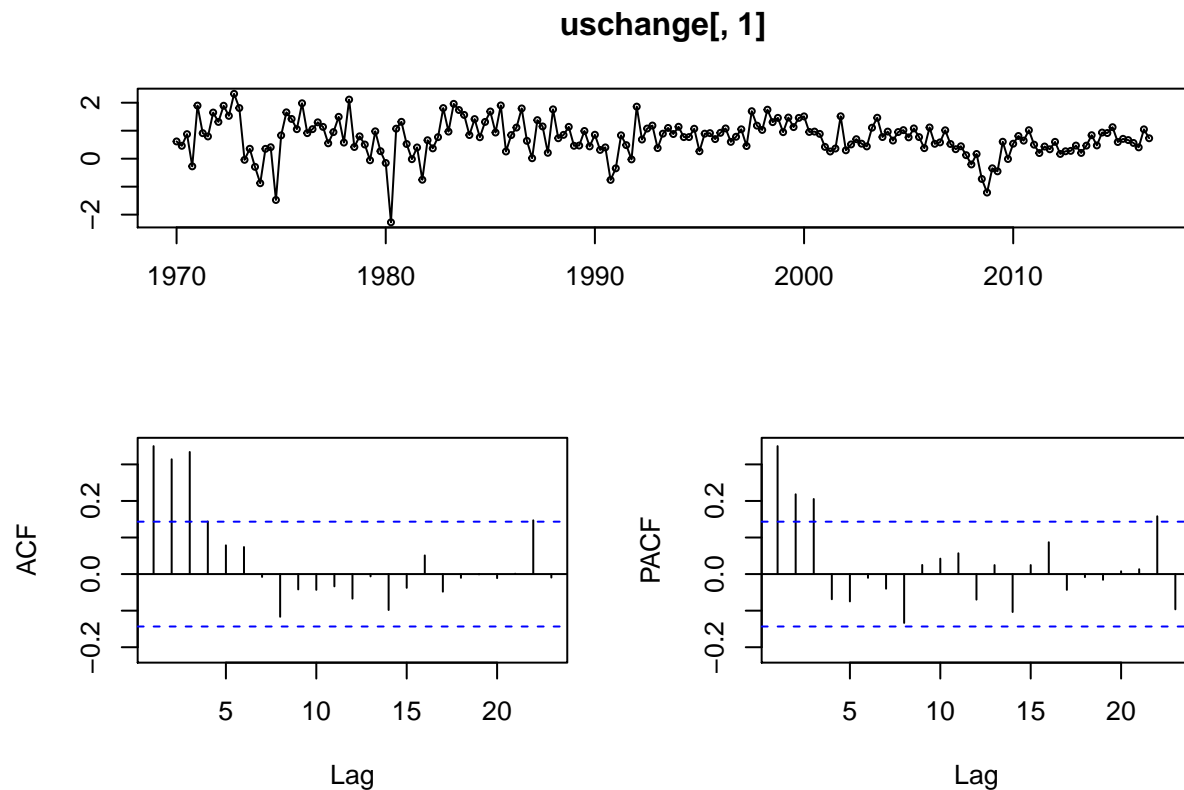
```
autoplot(uschange)
```



```
#Is the consumptio related to income? We expect a relation.
#Plots
par(mfrow=c(2,1))
#Consumption
plot(uschange[,1])
#Income
plot(uschange[,2])
```

```
par(mfrow=c(1,1))  
#More info on consumption variable  
tsdisplay(uschange[,1])
```

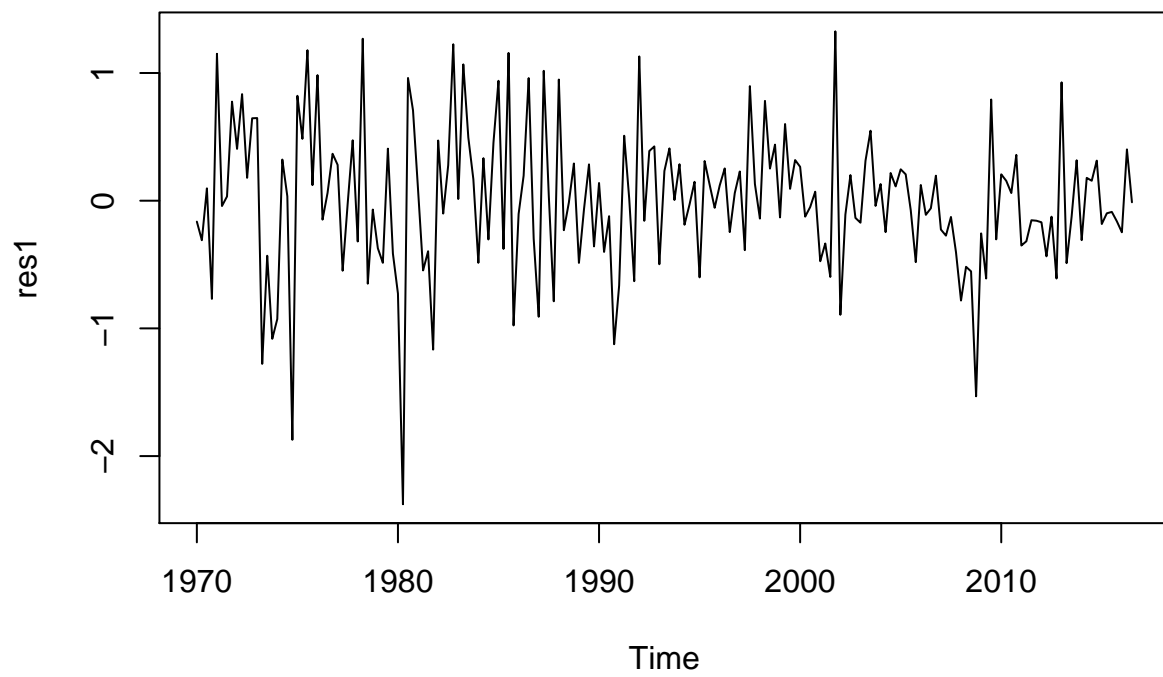


First ARMAX model “xreg” selects the regression terms (variables)

```
armax1<- Arima(uschange[,1], xreg=uschange[,2], order=c(1,0,1))
armax1
```

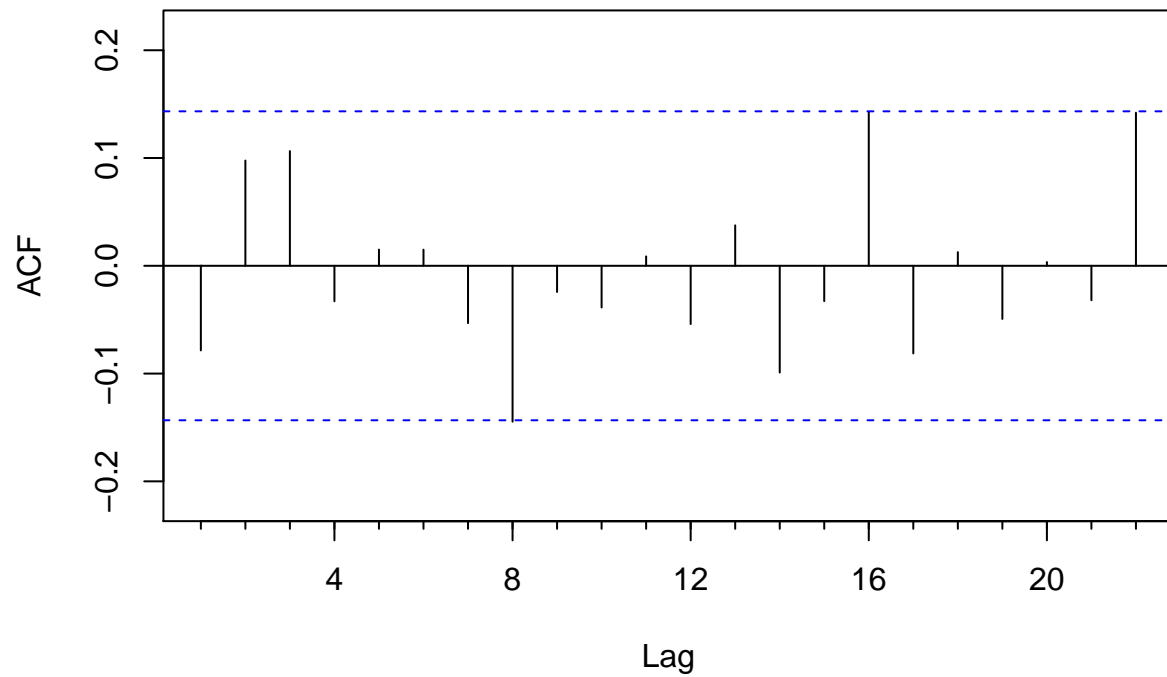
```
## Series: uschange[, 1]
## Regression with ARIMA(1,0,1) errors
##
## Coefficients:
##          ar1          ma1  intercept      xreg
##          0.7919   -0.5683      0.6133   0.1825
## s.e.    0.0809    0.1032      0.0912   0.0456
##
## sigma^2 estimated as 0.3302:  log likelihood=-159.81
## AIC=329.61   AICc=329.94   BIC=345.77
```

```
res1<- residuals(armax1)
#The residuals appear good but...
plot(res1)
```



`Acf(res1)`

Series res1

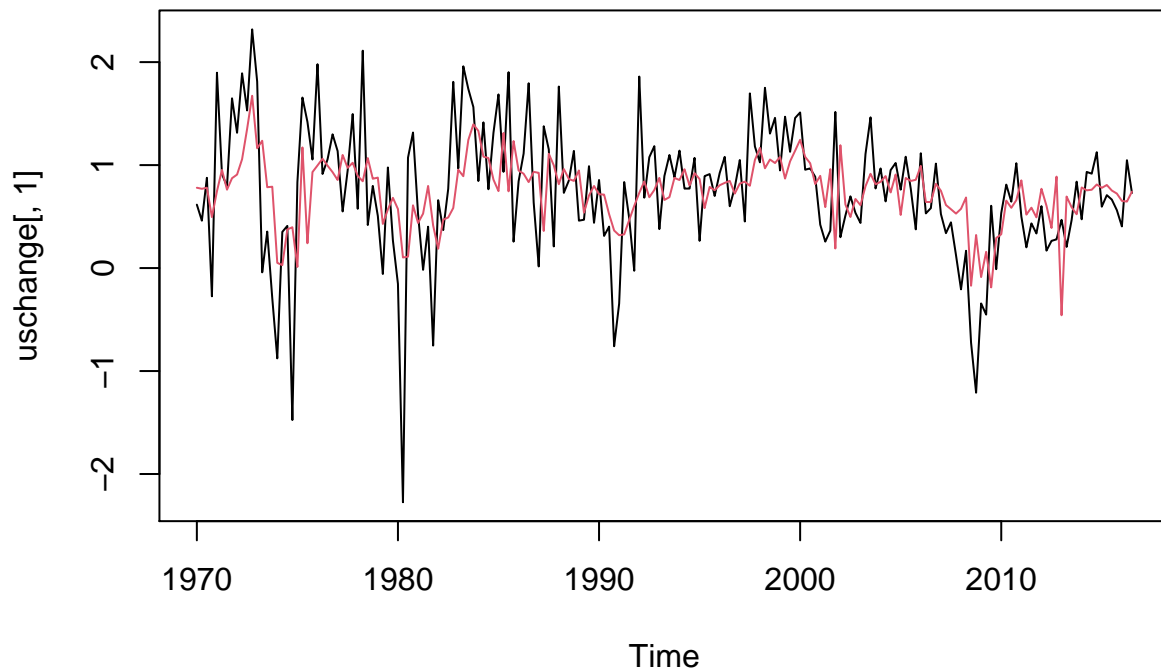


```
fitted(armax1)
```

##	Qtr1	Qtr2	Qtr3	Qtr4
## 1970	0.78006603	0.76998241	0.77997918	0.49470889
## 1971	0.74691458	0.95367578	0.76138513	0.87286648
## 1972	0.90802836	1.05689010	1.35176090	1.67290915
## 1973	1.16367516	1.23640518	0.78531105	0.79005897
## 1974	0.05003515	0.02887723	0.37542403	0.39653396
## 1975	0.01085034	1.17153692	0.24080247	0.93242178
## 1976	0.99640757	1.06214642	0.99472577	0.93136100
## 1977	0.85572449	1.09847602	0.97310973	1.02294306
## 1978	0.89498278	0.84279905	1.06793243	0.86730000
## 1979	0.87840579	0.42871159	0.56972244	0.68221667
## 1980	0.57340521	0.10300465	0.11068218	0.60940787
## 1981	0.43861630	0.52865939	0.79754365	0.41344059
## 1982	0.18783151	0.46954643	0.49131738	0.58362522
## 1983	0.95498317	0.89199407	1.24400153	1.39600580
## 1984	1.33190327	1.08208133	1.06911496	0.85908042
## 1985	0.74756500	1.31211568	0.74540820	1.23350168
## 1986	0.94718691	0.91606619	0.83474392	0.93429958
## 1987	0.92479494	0.36025112	1.11010775	0.99798195
## 1988	0.81430804	0.96157003	0.86476627	0.84641084
## 1989	0.94794685	0.53818285	0.70483443	0.79746028
## 1990	0.71661438	0.71355530	0.52369366	0.36479282
## 1991	0.31875176	0.32696302	0.46657727	0.60408691
## 1992	0.72976714	0.84119795	0.68756805	0.75856897

```
## 1993 0.87592653 0.66138154 0.68859617 0.87554555
## 1994 0.85504579 0.95994479 0.80156428 0.92224067
## 1995 0.86455278 0.58247944 0.78777694 0.75592063
## 1996 0.80440335 0.82728814 0.84656787 0.72326657
## 1997 0.82077487 0.83972711 0.79895963 1.04971381
## 1998 1.16710358 0.96916585 1.05446203 1.01986675
## 1999 1.07924160 0.86920134 1.03626172 1.13920607
## 2000 1.24577940 1.07983809 1.01776584 0.81608611
## 2001 0.89572256 0.59217313 0.96084875 0.19014878
## 2002 1.19322752 0.61764119 0.49594910 0.67128775
## 2003 0.61055054 0.79469592 0.91574479 0.81404964
## 2004 0.83743906 0.89386774 0.73441361 0.90861396
## 2005 0.51585232 0.87493303 0.84602049 0.85710258
## 2006 0.99228742 0.64174490 0.64139562 0.81848258
## 2007 0.75090681 0.61238757 0.57143827 0.52902716
## 2008 0.57538416 0.68538909 -0.17152326 0.32110519
## 2009 -0.08758067 0.15775900 -0.18758071 0.29165312
## 2010 0.32840865 0.65512112 0.58502461 0.65935684
## 2011 0.85147481 0.51911677 0.58684456 0.49275126
## 2012 0.77004410 0.60517415 0.38933485 0.88819759
## 2013 -0.45825324 0.69458103 0.59071556 0.52254602
## 2014 0.78259775 0.75487573 0.76063918 0.81156876
## 2015 0.77871811 0.80690024 0.75281487 0.72333875
## 2016 0.65205361 0.64574770 0.74192959
```

```
plot(uschange[,1])
lines(fitted(armax1), col=2)
```



The regression term appears significant ($0.1825/0.0456 > 2$) so it is useful to understand the behavior of our series. The choice of an ARIMA(1,0,1) derives from the fact that we have any trend (we have percentage change), instead the choice $p=1$ and $q=1$ are a careful way to find a possible structure for an ARIMA model (for a better selection see the guide) We see the fitted values (...the model doesn't capture in a good way the global behavior)

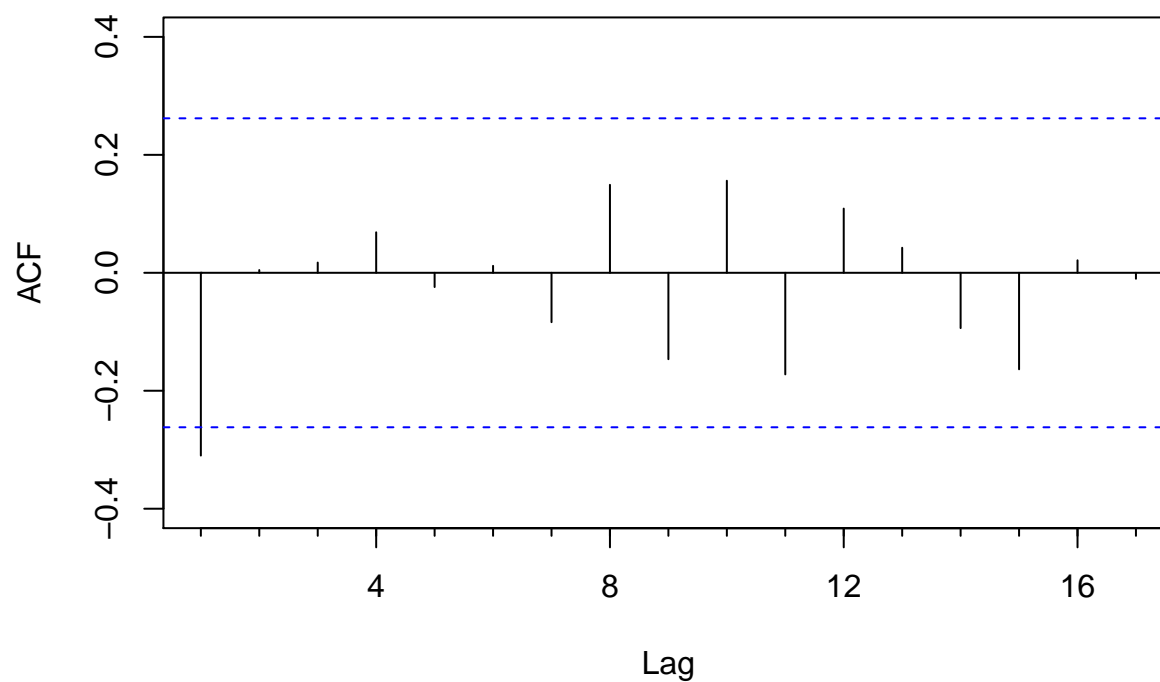
Second ARMAX model

```
armax2<- Arima(uschange[,1], xreg=uschange[,2], order=c(1,0,2))
armax2
```

```
## Series: uschange[, 1]
## Regression with ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1          ma1          ma2  intercept          xreg
##          0.6922      -0.5758      0.1984          0.5990      0.2028
## s.e.      0.1159      0.1301      0.0756          0.0884      0.0461
##
## sigma^2 estimated as 0.3219:  log likelihood=-156.95
## AIC=325.91   AICc=326.37   BIC=345.29
```

```
res2<- residuals(armax2)
#The first model is better
Acf(res)
```

Series res

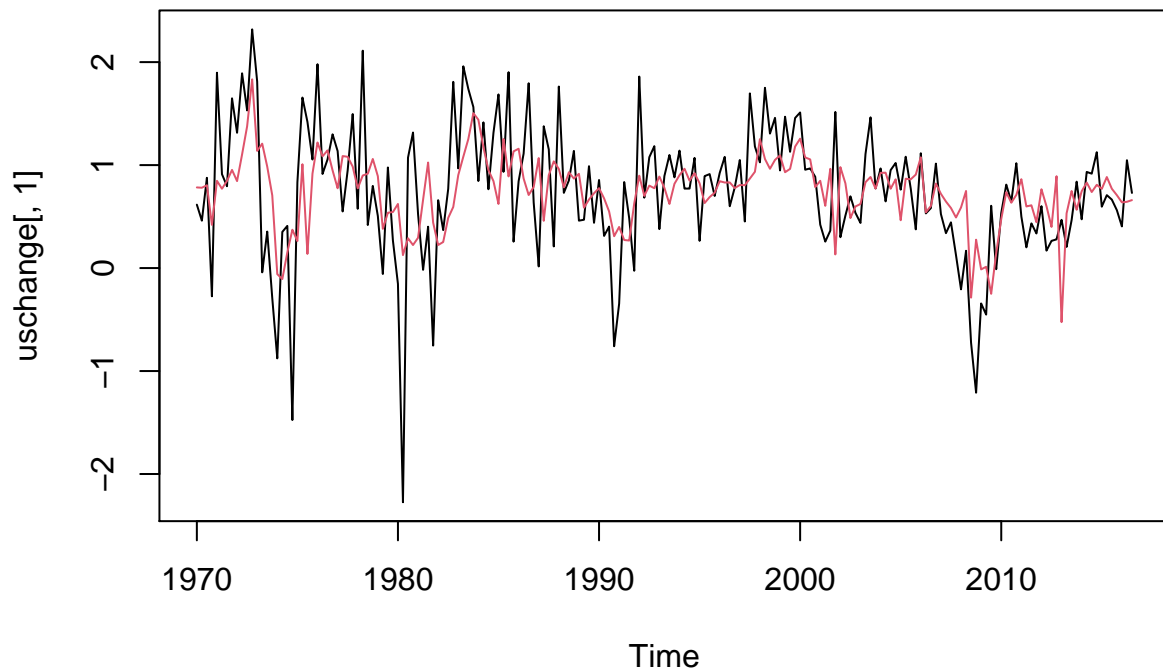


```
fitted(armax2)
```

##	Qtr1	Qtr2	Qtr3	Qtr4
## 1970	0.78312833	0.78018625	0.80479451	0.41930867
## 1971	0.84727727	0.77029485	0.84699950	0.95331366
## 1972	0.84462865	1.10359869	1.37654140	1.83317035
## 1973	1.13786583	1.20826089	0.98517520	0.70586675
## 1974	-0.05888372	-0.10652716	0.15252516	0.37252999
## 1975	0.26206496	1.00859262	0.13720674	0.91699737
## 1976	1.21948453	1.08521548	1.14267609	0.94879507
## 1977	0.77480045	1.08771984	1.08065070	0.98559686
## 1978	0.77381614	0.89892787	0.91640998	1.05941956
## 1979	0.89261857	0.38091069	0.53759910	0.54387460
## 1980	0.62276776	0.12597681	0.28963813	0.22350695
## 1981	0.29439649	0.66462244	1.02513172	0.44125915
## 1982	0.22393882	0.25312328	0.48707851	0.59336917
## 1983	0.90313806	1.08127964	1.25264967	1.50432058
## 1984	1.44036881	1.17436646	0.95189308	0.84432657
## 1985	0.62302160	1.25286288	0.88979506	1.13370425
## 1986	1.15901792	0.86946926	0.70947373	0.78744003
## 1987	1.06836399	0.45881327	0.90243930	1.03659542
## 1988	0.96270927	0.78370422	0.93047644	0.86924854
## 1989	0.91533693	0.58627680	0.66536135	0.72658394
## 1990	0.77608713	0.67685037	0.55014995	0.30925675
## 1991	0.39935061	0.27248164	0.26732620	0.62816783
## 1992	0.89742511	0.70072700	0.80056755	0.77549795

##	1993	0.88877001	0.77843395	0.62254092	0.81818587
##	1994	0.90212475	0.96475852	0.84927848	0.92193247
##	1995	0.82302821	0.63143690	0.68951971	0.73146966
##	1996	0.84367658	0.83076836	0.83169021	0.77671808
##	1997	0.80639086	0.80452612	0.86904546	0.93489503
##	1998	1.25190643	1.05908067	0.96434947	1.05049178
##	1999	1.09414605	0.93073127	0.96307717	1.17990917
##	2000	1.25691761	1.07448753	1.05545026	0.78640224
##	2001	0.84779060	0.60439534	0.96407640	0.13244226
##	2002	0.98024037	0.82330683	0.48567225	0.59802581
##	2003	0.62215476	0.83382918	0.88479540	0.77211433
##	2004	0.92453660	0.92649952	0.77051709	0.86179287
##	2005	0.46441540	0.86776267	0.86193275	0.90781283
##	2006	1.07577457	0.54136888	0.60248018	0.82077639
##	2007	0.72101375	0.64631554	0.58322422	0.49104710
##	2008	0.58814023	0.74866969	-0.28741575	0.27567649
##	2009	-0.01283404	0.01246222	-0.24998664	0.14688615
##	2010	0.47625237	0.73744143	0.63427916	0.71035984
##	2011	0.86308493	0.59824770	0.60961002	0.44522796
##	2012	0.76540887	0.60120651	0.39972191	0.89227477
##	2013	-0.52514212	0.53160833	0.74818031	0.56648441
##	2014	0.74603201	0.83062237	0.73814649	0.80864084
##	2015	0.76900278	0.88423189	0.77020729	0.70789375
##	2016	0.63785994	0.64367730	0.65987214	

```
plot(uschange[,1])
lines(fitted(armax2), col=2)
```

The model doesn't capture in a good way the global behavior also in this case (but is better than the previous one)

To select the model, considering the fact that are very similar (the first has better residuals behavior, the second fits better) we use AIC

```
AIC(arima1)
```

```
## [1] 342.7583
```

```
AIC(armax2)
```

```
## [1] 325.908
```

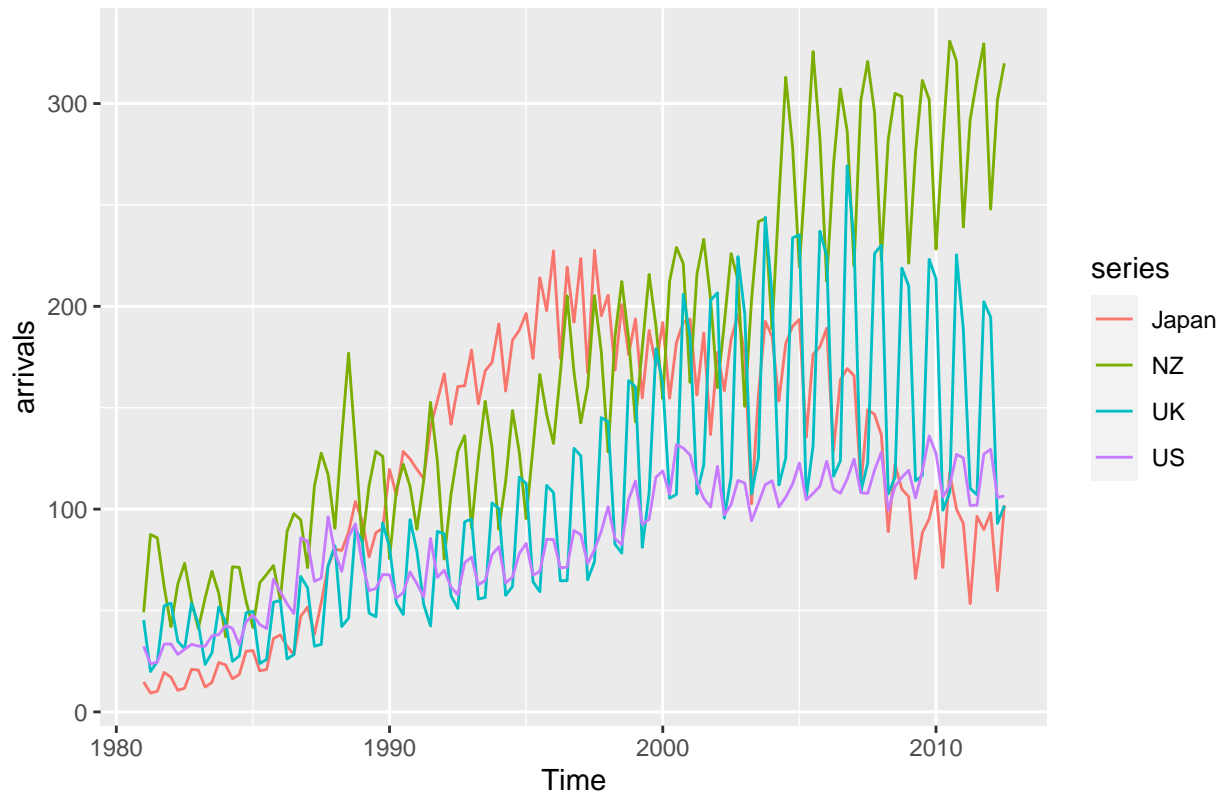
We prefer the second model. We want lower AIC

Procedure also available with `auto.arima`

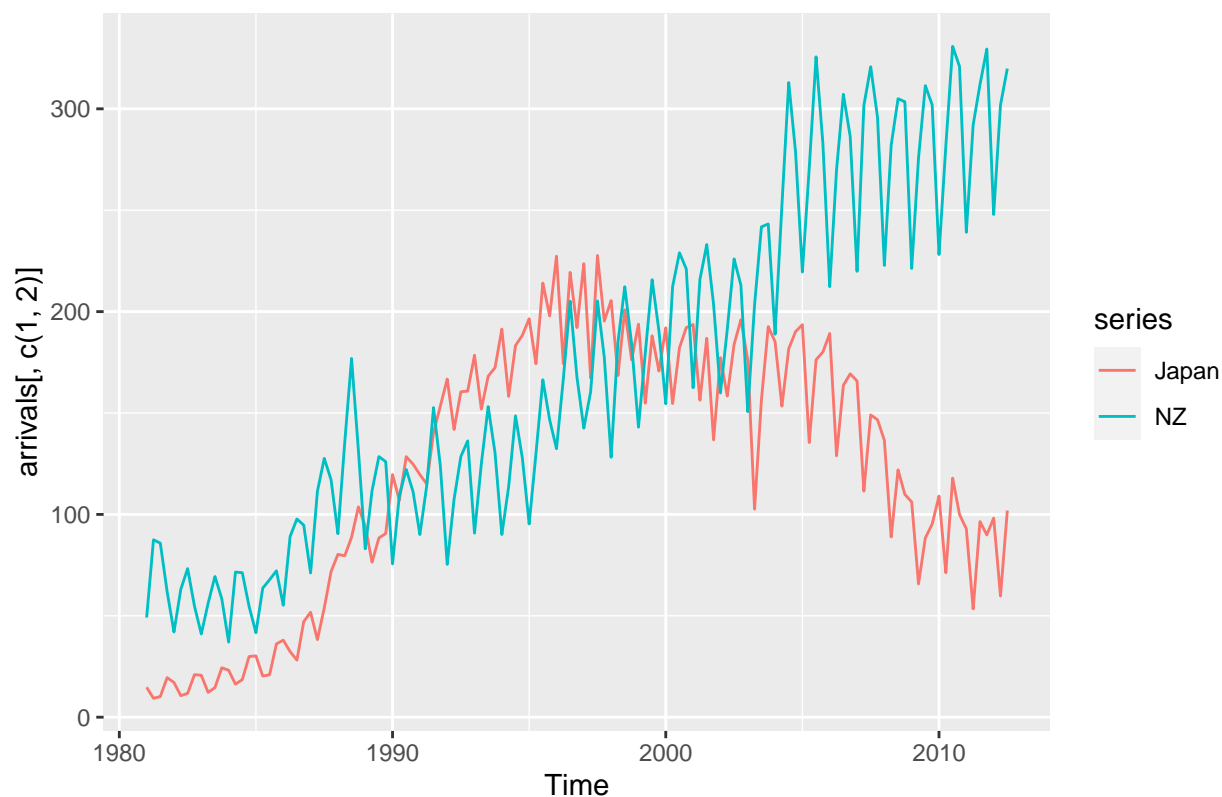
```
auto.arima<- auto.arima(uschange[,1], xreg=uschange[,2])
```

Quarterly international arrivals (in thousands) to Australia from Japan, New Zealand, UK and the US. 1981Q1 - 2012Q3

```
autoplot(arrivals)
```



```
autoplot(arrivals[,c(1,2)])
```



All the variables have seasonality and different trends. Japan has a strongly NON-linear behavior, instead for New Zealand is LINEAR. We focus on Japan and New Zealand because they start in the same way but then from 2001 they have opposite trends. We can hypothesize a competitive behavior for the two countries

```
#Variables definition
```

```
Japan<- arrivals[,1]
NZ<- arrivals[,2]
UK<- arrivals[,3]
US<- arrivals[,4]
```

We try with a simple arima model (not reasonable forecast). We prefer a linear model in this case. The dependent variable is NZ and regression variable is Japan.

```
auto.arima<- auto.arima(NZ, xreg=Japan)
auto.arima
```

```
## Series: NZ
## Regression with ARIMA(2,1,1)(0,1,1)[4] errors
##
## Coefficients:
##          ar1      ar2      ma1      sma1      xreg
##          0.6117  0.1880 -0.9874 -0.5406  0.1768
## s.e.      0.0950  0.0934   0.0498   0.0911  0.0873
##
## sigma^2 estimated as 157:  log likelihood=-480.81
## AIC=973.61   AICc=974.34   BIC=990.43
```

We try with a regression model with trend, season and external variable 'Japan'

```
mod<- tslm(NZ~ trend+season+Japan)
#The significance and the negative sign confirms the presence of a competitive behavior
summary(mod)
```

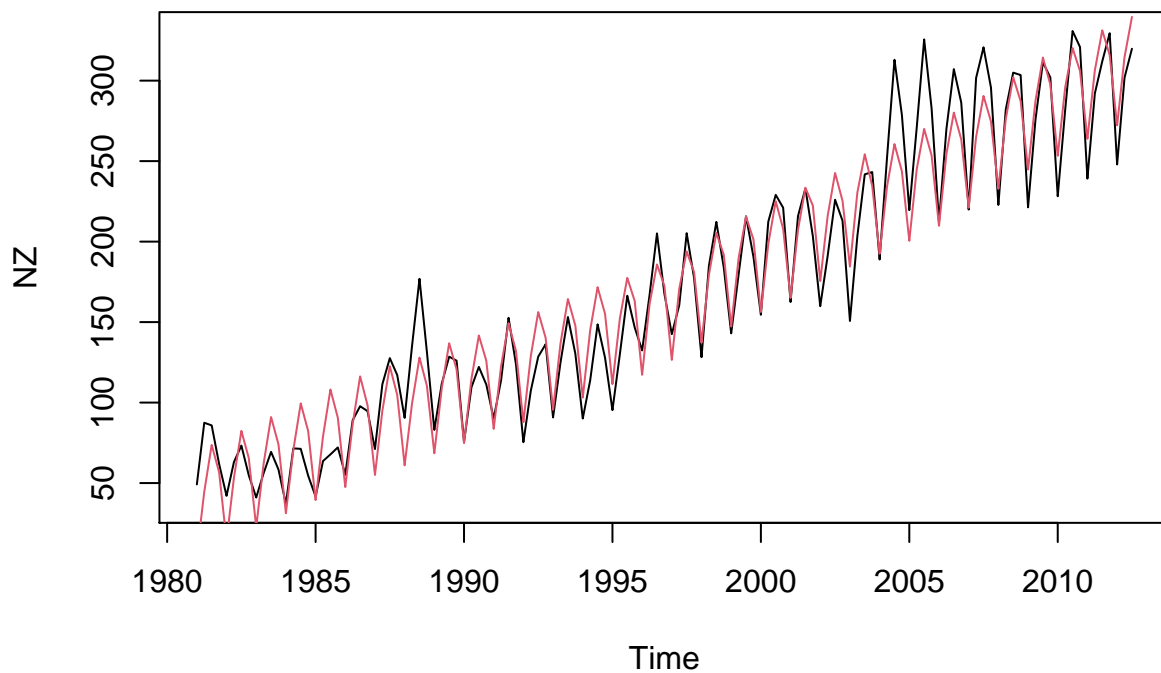
```
##
## Call:
## tslm(formula = NZ ~ trend + season + Japan)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -40.32 -14.12  -2.51   12.70   55.63
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.73757    5.21347   0.909  0.36531
## trend        2.21843    0.05539  40.052 < 2e-16 ***
## season2     36.35773    5.01769   7.246 4.38e-11 ***
## season3     63.33026    4.92212  12.866 < 2e-16 ***
## season4     45.09142    4.95892   9.093 2.33e-15 ***
## Japan       -0.10039    0.03232  -3.106 0.00236 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.67 on 121 degrees of freedom
## Multiple R-squared:  0.9482, Adjusted R-squared:  0.946
## F-statistic: 442.9 on 5 and 121 DF, p-value: < 2.2e-16
```

```
fitted(mod)
```

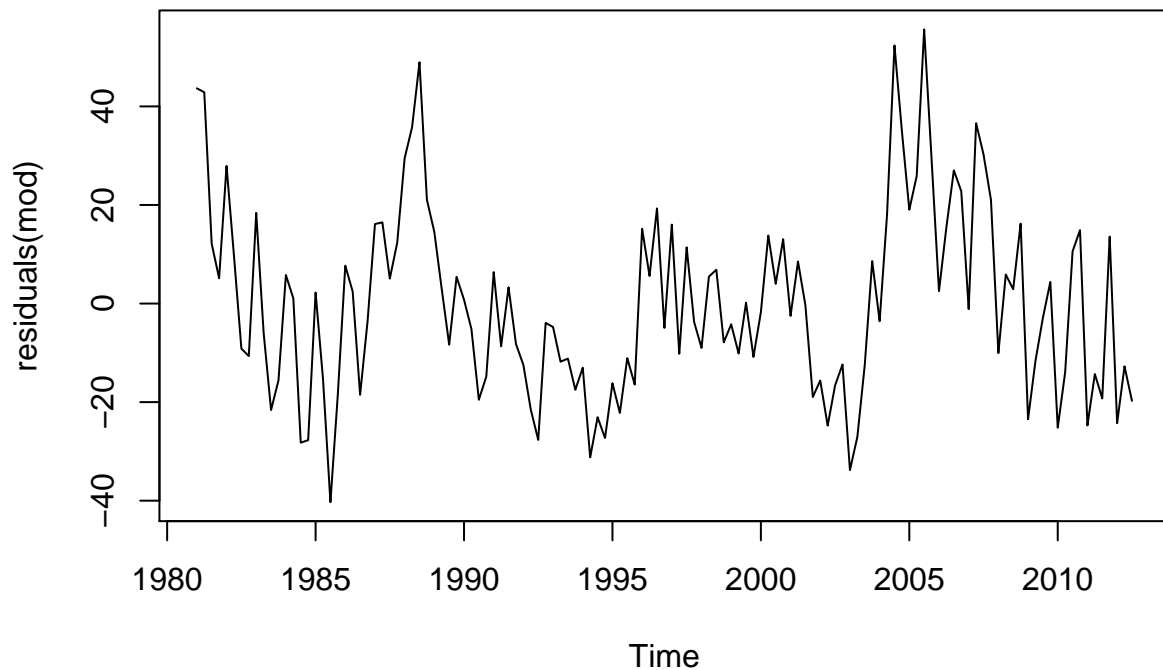
```
##           Qtr1           Qtr2           Qtr3           Qtr4
## 1981  5.473922  44.596412  73.702533  56.744161
## 1982 14.111305  53.340009  82.418522  65.472097
## 1983 22.628218  62.051280  91.008119  74.004269
## 1984 31.251144  70.517295  99.486582  82.318292
## 1985 39.414981  78.991039 108.118945  90.566457
## 1986 47.510752  86.651414 116.267522  98.337764
## 1987 55.004376  94.934013 122.563578 104.736118
## 1988 61.010502  99.657338 127.933523 110.404225
## 1989 68.491577 108.845668 136.838850 120.604200
## 1990 74.807108 114.657135 141.688969 126.039901
## 1991 83.682418 122.719275 149.358178 132.036088
## 1992 87.828292 128.898776 156.225560 140.169807
## 1993 95.524004 136.766759 164.328660 147.883890
## 1994 103.102559 145.007697 171.680632 155.159866
## 1995 111.462962 152.265100 177.464491 163.072122
## 1996 117.239090 161.136997 185.808128 172.519362
## 1997 126.483740 170.719062 193.849186 181.070508
## 1998 137.181756 179.467177 205.411471 191.863193
## 1999 147.224618 189.712227 215.568177 201.293668
## 2000 156.278926 198.601893 225.039210 208.014379
## 2001 164.989795 207.314770 233.437962 222.448459
```

```
## 2002 175.508110 215.982270 242.616856 225.373172
## 2003 184.495357 230.450182 254.243793 234.589712
## 2004 192.461222 234.231033 260.581310 243.709174
## 2005 200.494450 244.901943 269.998834 253.592614
## 2006 209.805860 254.432307 280.136766 263.547634
## 2007 221.025108 265.056434 290.474578 274.694299
## 2008 232.838170 276.199485 302.070293 287.263910
## 2009 244.765880 287.395341 314.315139 297.607746
## 2010 253.343530 295.716391 320.226796 306.002282
## 2011 263.841867 306.382683 331.249779 315.888633
## 2012 272.184400 314.617597 339.578056
```

```
plot(NZ)
lines(fitted(mod), col=2)
```



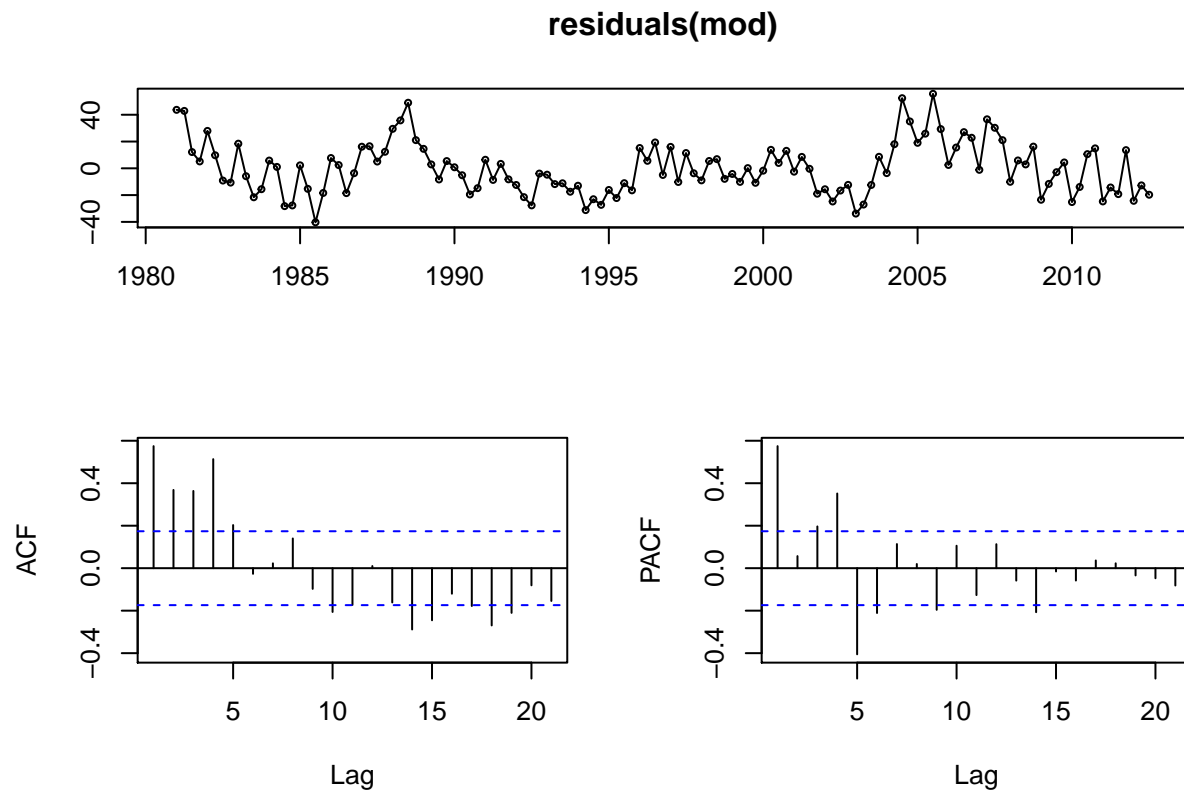
```
#With an "armonic" behavior of residuals we can say that we may be in presence of positive
#autocorrelation in residuals
plot(residuals(mod))
```



```
#Analysis of residuals: are there autocorrelation?
dw<- dwtest(mod, alt="two.sided")
#This result confirm that we re dealing with autocorrelation in residuals (the autoc. is not 0)
dw
```

```
##
## Durbin-Watson test
##
## data: mod
## DW = 0.80163, p-value = 1.074e-11
## alternative hypothesis: true autocorrelation is not 0
```

```
#We see that there is something that remains unexplained in our data. We don't capture all the
#informations in our data
tsdisplay(residuals(mod))
```



```
#Fit an arima model to residuals
aar<- auto.arima(residuals(mod))
#It is a quite complex model (many parameters)
aar
```

```
## Series: residuals(mod)
## ARIMA(2,0,0)(1,1,2)[4]
##
## Coefficients:
##          ar1      ar2      sar1      sma1      sma2
##          0.5854  0.1708 -0.6487  0.1768 -0.5394
## s.e.    0.0895  0.0903   0.1969  0.1792   0.0995
##
## sigma^2 estimated as 159.2:  log likelihood=-485.03
## AIC=982.07   AICc=982.79   BIC=998.94
```

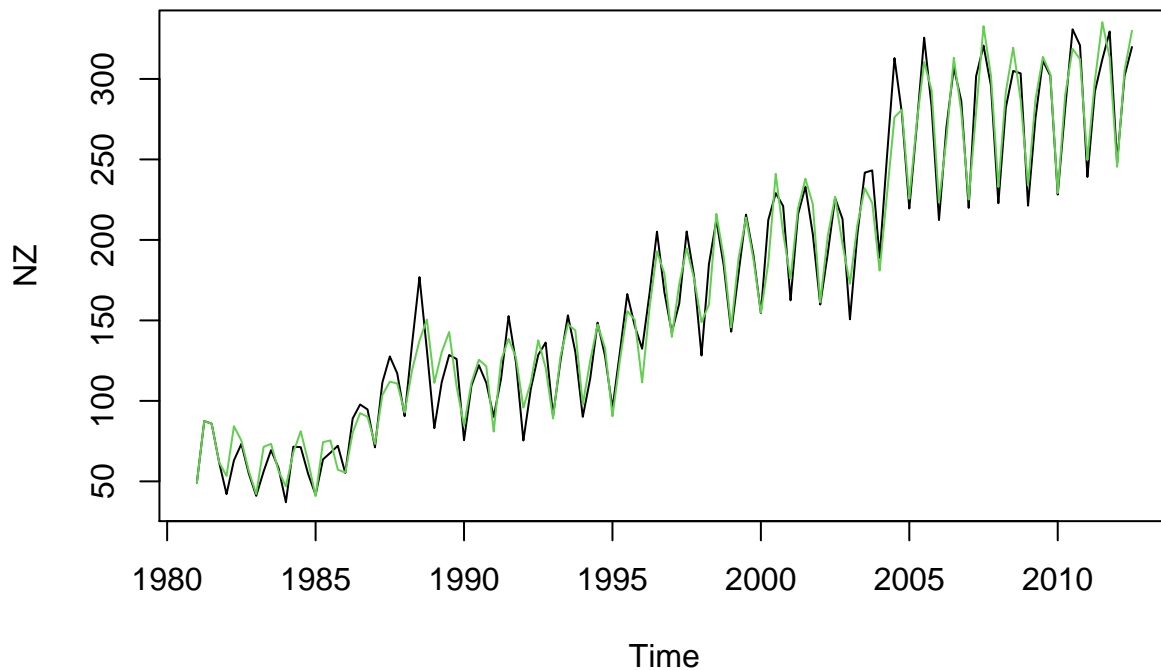
```
fitted(aar)
```

```
##           Qtr1           Qtr2           Qtr3           Qtr4
## 1981  43.62241191  42.82771716  12.12632859   5.13270118
## 1982  39.25685005  30.94381916  -7.09550602  -9.03874815
## 1983  19.51340004   9.39816502 -17.69402943 -17.44833143
## 1984  15.60740629 -2.53417875 -18.35382004 -19.94140170
## 1985   1.42372037 -4.59667453 -32.70369501 -33.30637176
## 1986   7.94093963 -6.37015741 -23.87810402  -8.22195090
```

```
## 1987 18.20485704 8.71776054 -10.64261642 6.10437564
## 1988 31.75361690 19.40986265 9.55563715 40.07433784
## 1989 42.74944577 21.49510026 5.93845976 -11.58506339
## 1990 9.75089423 -3.28406989 -16.10793367 -4.56096181
## 1991 -2.71733325 2.60100116 -10.84796350 -5.02720168
## 1992 8.00788956 -17.38465325 -18.62417361 -19.57522132
## 1993 -6.47962136 -9.03156986 -16.75772101 -4.09810636
## 1994 -5.35000150 -20.11215353 -24.23849355 -22.42976111
## 1995 -21.02495330 -27.23822955 -21.70574669 -12.67371399
## 1996 -5.77052687 -3.25150840 7.08363833 6.57020050
## 1997 13.28416382 1.97236765 0.95055894 -5.11615174
## 1998 11.76561337 -19.71404711 10.78224507 -2.79099272
## 1999 -1.53430708 -0.94421897 -1.66864894 -13.39790786
## 2000 -1.55342216 -12.92139587 15.98152344 -3.64814663
## 2001 10.84150745 12.19287382 4.55358952 -0.21974557
## 2002 -14.25959752 -13.90644122 -15.75780959 -27.95211138
## 2003 -11.72201908 -21.15214421 -22.10433630 -11.59198901
## 2004 -11.47947223 -5.65630402 15.69245611 37.14157651
## 2005 25.06587872 27.70200275 40.52727819 39.46423518
## 2006 13.46662746 8.56477311 33.01386681 15.94767954
## 2007 4.05675646 14.30619067 42.31821071 27.20343865
## 2008 0.09184271 15.47606318 17.34473211 -0.29834327
## 2009 -11.07669839 -0.17670216 -0.59439378 5.41668433
## 2010 -24.27319699 -5.76879111 -1.51249089 6.31922431
## 2011 -14.22855235 -7.60442828 4.01320823 -2.52525146
## 2012 -26.77243162 -9.46606687 -9.63470544
```

Complete the analysis by summing predictions made with linear model and ARIMA on residuals. Combination of models seen also in Lab 1

```
plot(NZ)
lines(fitted(mod)+fitted(aar), col=3)
```

Another way of performing the same linear regression tt: time for modelling the trend

```
tt<- (1:length(NZ))
#seas factorizes the observations
#1:3: adds the last 3 observations (without this, "rep" deletes them)
seas <- factor(c(rep(1:4,length(NZ)/4),1:3))
#All the parameter are linear
mod2 <- lm(NZ~ tt+seas+Japan)
#We have the same results of mod (with tslm function)
summary(mod2)
```

```
##
## Call:
## lm(formula = NZ ~ tt + seas + Japan)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -40.32 -14.12  -2.51  12.70  55.63
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.73757    5.21347   0.909  0.36531
## tt            2.21843    0.05539  40.052 < 2e-16 ***
## seas2         36.35773    5.01769   7.246 4.38e-11 ***
## seas3         63.33026    4.92212  12.866 < 2e-16 ***
## seas4         45.09142    4.95892   9.093 2.33e-15 ***
```

```
## Japan      -0.10039    0.03232  -3.106  0.00236 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.67 on 121 degrees of freedom
## Multiple R-squared:  0.9482, Adjusted R-squared:  0.946
## F-statistic: 442.9 on 5 and 121 DF,  p-value: < 2.2e-16
```

```
AIC(mod2)
```

```
## [1] 1124.947
```

```
AIC(mod)
```

```
## [1] 1124.947
```

GAM model

Are the parameters all linear? To see if `tt` and `Japan` has a NON-linear role we use GAM. `s()` stands for smoothing spline. Values for `df` should be greater than 1, with `df=1` implying a linear fit. In our model we say that the `df` could be 2,3 or 4 (increasingly linear structure). More high is the `df` more jumpier is the function.

```
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.20
```

```
g1 <- gam(NZ~s(tt)+seas+s(Japan),arg=c("df=2","df=3","df=4"))
summary(g1)
```

```
##
## Call: gam(formula = NZ ~ s(tt) + seas + s(Japan), arg = c("df=2", "df=3",
##      "df=4"))
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -33.2898 -11.2819  -0.6276  10.1972  44.1405
##
## (Dispersion Parameter for gaussian family taken to be 250.3224)
##
##      Null Deviance: 903510.7 on 126 degrees of freedom
## Residual Deviance: 28787.11 on 115.0002 degrees of freedom
## AIC: 1075.194
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
```

```
##           Df Sum Sq Mean Sq F value    Pr(>F)
## s(tt)      1 712377  712377 2845.84 < 2.2e-16 ***
## seas       3  71867   23956   95.70 < 2.2e-16 ***
## s(Japan)    1  31162   31162  124.49 < 2.2e-16 ***
## Residuals 115  28787     250
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##           Npar Df Npar F      Pr(F)
## (Intercept)
## s(tt)              3 31.059 8.882e-15 ***
## seas
## s(Japan)           3 10.977 2.154e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

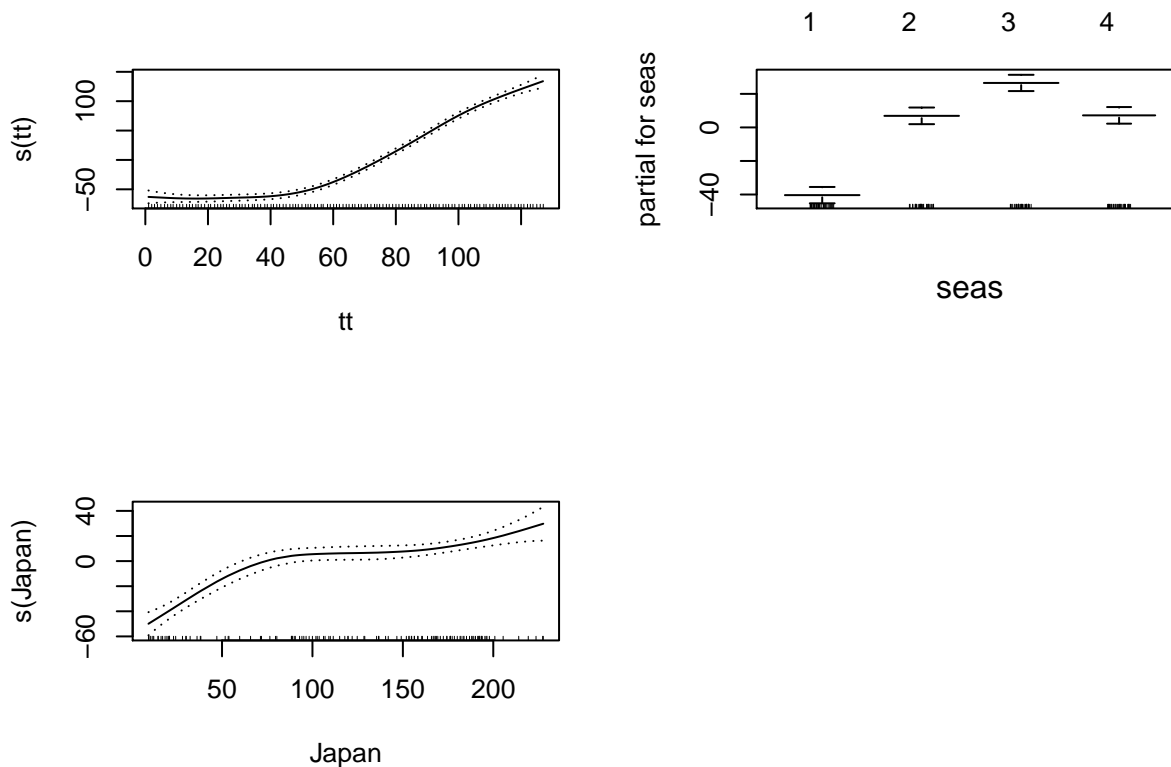
We have two ANOVA: for parametric and NON-parametric effects. For seas we don't have a non-par. part because we don't model it with splines (it is pointless modelling the seasonality with smoothing splines because it is a factorial variable).

```
#Time and Japan have a nonlinear effect
par(mfrow=c(2,2))
#These plots confirm the fact that tt and Japan have a non-linear relation with NZ
plot(g1, se=T)
#GAM is the best model
AIC(mod2)
```

```
## [1] 1124.947
```

```
AIC(g1)
```

```
## [1] 1075.194
```



```
#Try another option with loess (lo)
g2<- gam(NZ~lo(tt)+seas+lo(Japan))
```

```
## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, :
## lo.wam convergence not obtained in 30 iterations
```

```
## Warning in lo.wam(x, z, wz, fit$smooth, which, fit$smooth.frame, bf.maxit, :
## lo.wam convergence not obtained in 30 iterations
```

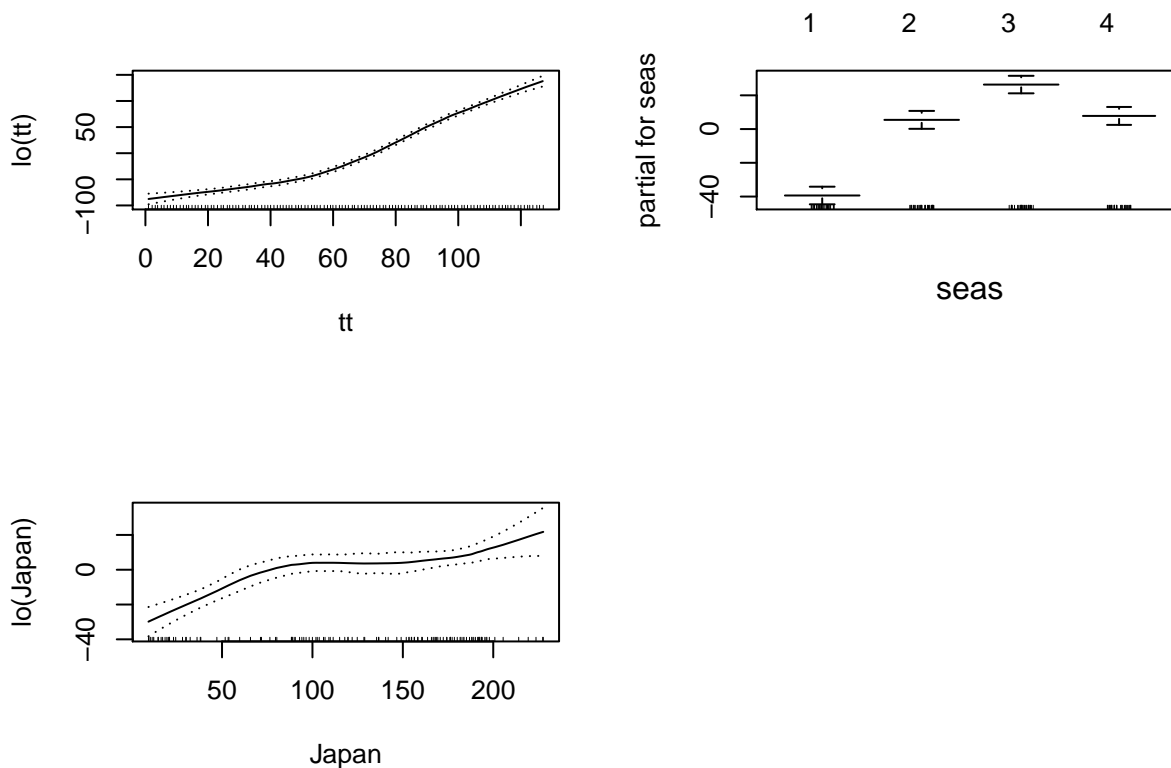
```
summary(g2)
```

```
##
## Call: gam(formula = NZ ~ lo(tt) + seas + lo(Japan))
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -35.3030 -11.4913  -0.1418  10.5378  46.9110
##
## (Dispersion Parameter for gaussian family taken to be 287.9386)
##
##      Null Deviance: 903510.7 on 126 degrees of freedom
## Residual Deviance: 33432.97 on 116.1115 degrees of freedom
## AIC: 1091.973
##
## Number of Local Scoring Iterations: NA
```

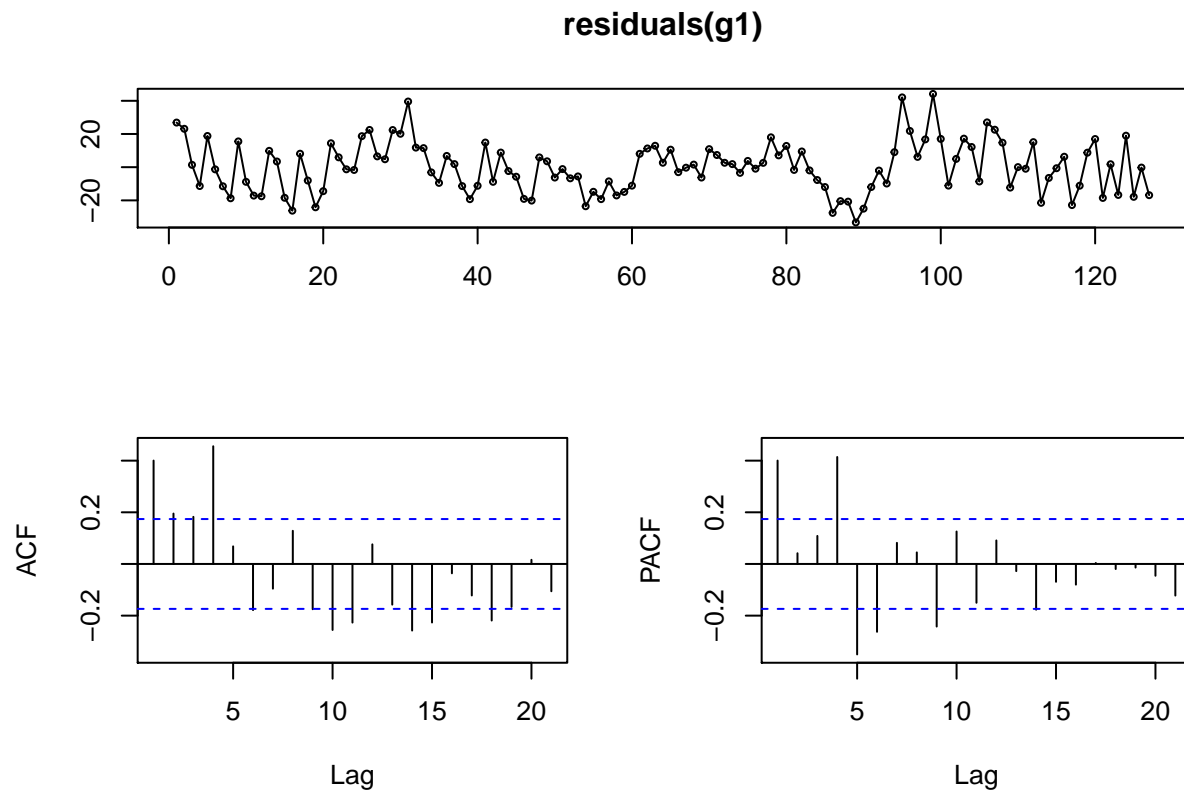
```
##
## Anova for Parametric Effects
##           Df Sum Sq Mean Sq  F value    Pr(>F)
## lo(tt)      1.00 741830   741830 2576.347 < 2.2e-16 ***
## seas        3.00  71110    23703   82.321 < 2.2e-16 ***
## lo(Japan)    1.00  12565    12565   43.637 1.25e-09 ***
## Residuals 116.11  33433      288
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##           Npar Df  Npar F      Pr(F)
## (Intercept)
## lo(tt)          2.4 21.7685 1.114e-09 ***
## seas
## lo(Japan)       2.5  5.1328  0.00387 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow=c(2,2))
plot(g2, se=T)
#It's better then mod2 but worse then g1
AIC(g2)
```

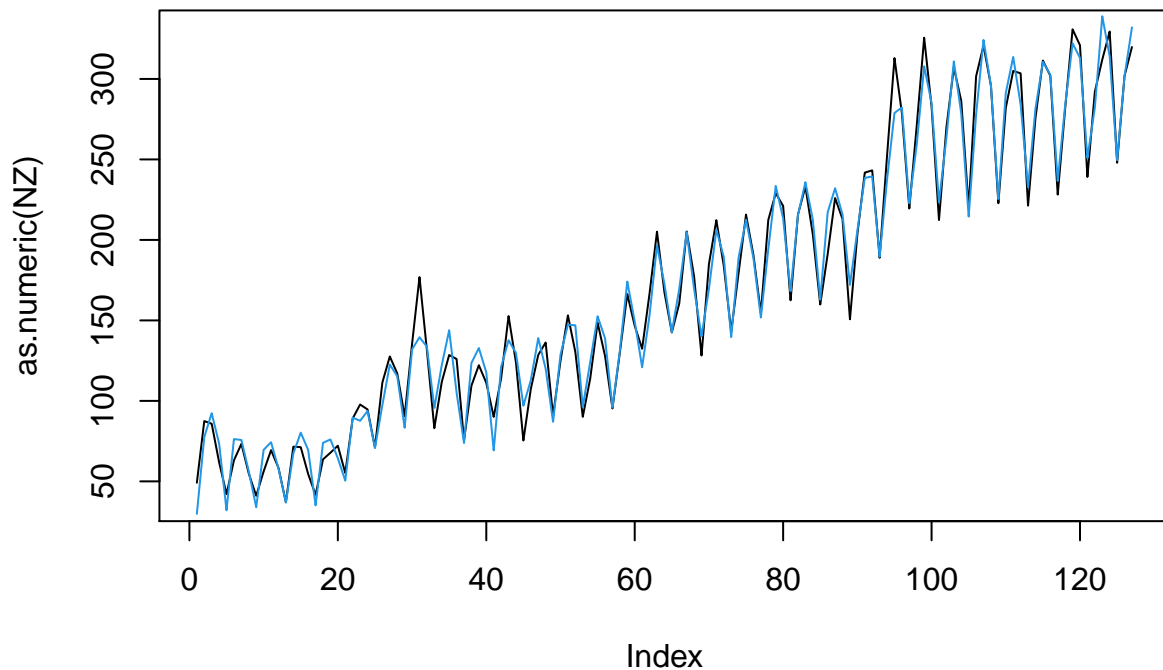
```
## [1] 1091.973
```



```
#Perform an analysis of residuals
tsdisplay(residuals(g1))
```



```
aar1<- auto.arima(residuals(g1))
plot(as.numeric(NZ), type="l")
#Combination of g1 on data and Arima on residuals
lines(fitted(aar1)+ fitted(g1), col=4)
```

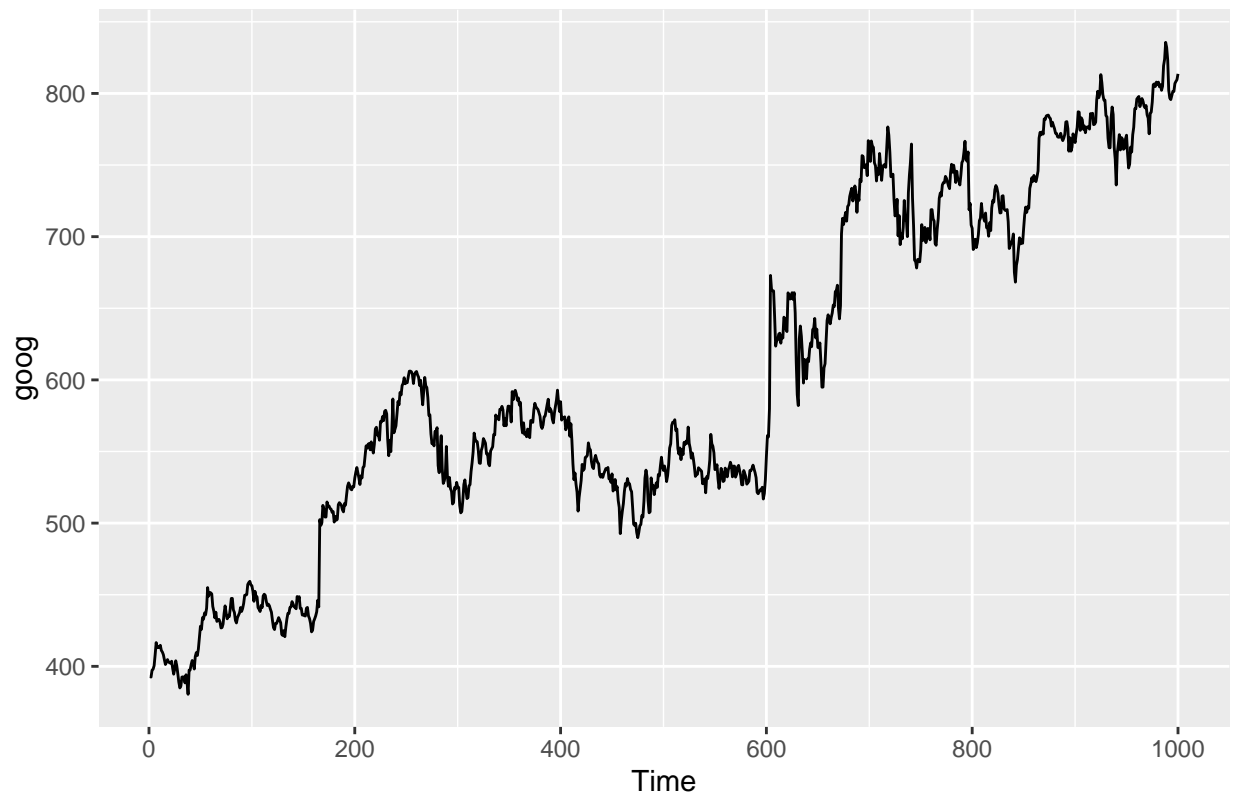


We don't make any forecast because it is available only if we have future values of 'Japan', general problem of this approach. A possible solution is model the t.s. of Japan (for example with a BASS model) and make some forecasts and use them to make forecast on NZ with our initial model. We can use a BASS model because the data are turistic destinations, so they are a product and follow a cycle product pattern

Exercise 6: model Japan variable with BASS model and use its forecasts to make predictions with our initial model

Google stock price: Daily closing stock prices of Google Inc

```
autoplot(goog)
```



CASE STUDY LAB 2

#Dataset and preliminary analysis

```
dati <- read.csv("movies.csv", stringsAsFactors=TRUE)
str(dati)
```

```
## 'data.frame':  3229 obs. of  27 variables:
## $ X.1          : int  1 2 3 4 5 6 7 8 9 10 ...
## $ X            : int  1 2 3 4 5 6 7 8 9 10 ...
## $ budget       : int  237000000 300000000 245000000 250000000 260000000 258000000 260000000 ...
## $ popularity   : num  1.50e+08 1.39e+08 1.07e+08 1.12e+07 4.39e+07 ...
## $ production_countries: int  1 1 1 1 1 1 1 1 1 1 ...
## $ release_date  : Factor w/ 2494 levels "01/01/1961","01/01/1969",...: 785 1552 2190 1304 470 ...
## $ revenue       : num  2.79e+09 9.61e+08 8.81e+08 1.08e+09 2.84e+08 ...
## $ runtime       : int  162 169 148 165 132 139 100 141 153 151 ...
## $ spoken_languages : int  1 1 1 1 1 1 1 1 1 1 ...
## $ vote_average   : num  7.2 6.9 6.3 7.6 6.1 5.9 7.4 7.3 7.4 5.7 ...
## $ vote_classes   : Factor w/ 5 levels "0 a 5","5 a 6",...: 4 3 3 4 3 2 4 4 4 2 ...
## $ Comp_Universal : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Paramount : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Warner    : int  0 0 0 1 0 0 0 0 1 1 ...
## $ Comp_20fox     : int  1 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Columbia  : int  0 0 1 0 0 1 0 0 0 0 ...
```



```
## $ Comp_NewLine      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Disney       : int  0 1 0 0 1 0 1 0 0 0 ...
## $ Comp_VillageRoadshow: int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Miramax      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Pixar        : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_RelMedia     : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_MGM          : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_DreamWorks   : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Touchstone   : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_CanalPlus    : int  0 0 0 0 0 0 0 0 0 0 ...
## $ genere            : Factor w/ 19 levels "Action","Adventure",...: 9 2 1 1 1 1 3 1 9 1 ...
```

```
#Srased columns of indicator variables (useless)
dati<-dati[,-c(1,2)]
```

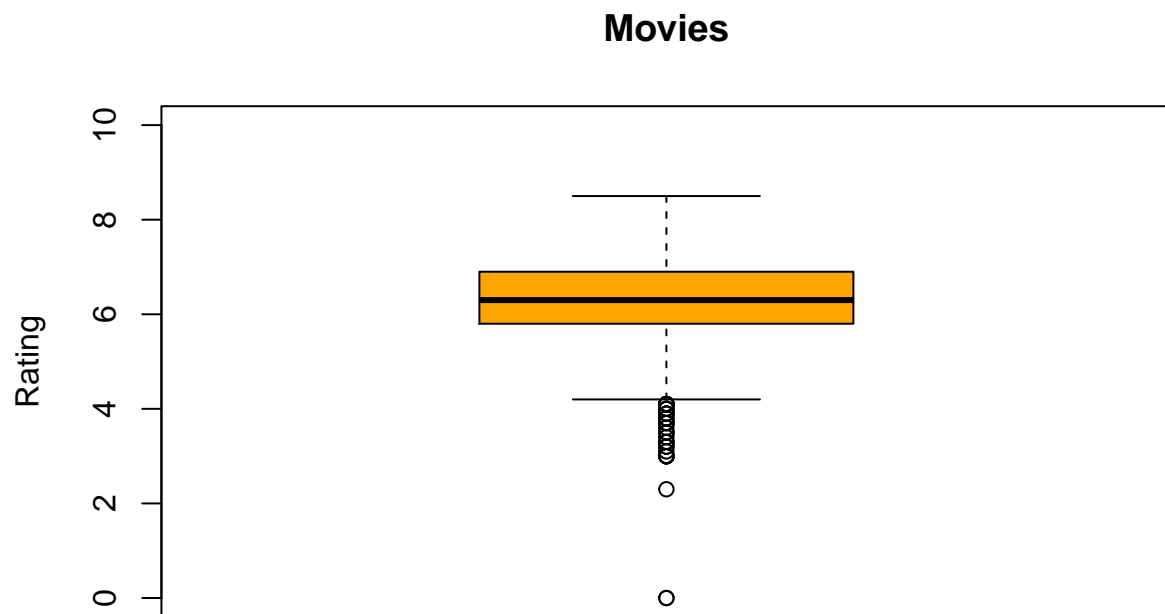
```
#Transform variable release_date in format "data"
dati$release_date <- as.Date(dati$release_date, "%d/%m/%Y")
str(dati)
```

```
## 'data.frame':    3229 obs. of  25 variables:
## $ budget          : int  237000000 300000000 245000000 250000000 260000000 258000000 260000000 ...
## $ popularity      : num  1.50e+08 1.39e+08 1.07e+08 1.12e+07 4.39e+07 ...
## $ production_countries: int  1 1 1 1 1 1 1 1 1 1 ...
## $ release_date    : Date, format: "2009-12-10" "2007-05-19" ...
## $ revenue         : num  2.79e+09 9.61e+08 8.81e+08 1.08e+09 2.84e+08 ...
## $ runtime         : int  162 169 148 165 132 139 100 141 153 151 ...
## $ spoken_languages : int  1 1 1 1 1 1 1 1 1 1 ...
## $ vote_average    : num  7.2 6.9 6.3 7.6 6.1 5.9 7.4 7.3 7.4 5.7 ...
## $ vote_classes     : Factor w/ 5 levels "0 a 5","5 a 6",...: 4 3 3 4 3 2 4 4 4 2 ...
## $ Comp_Universal   : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Paramount   : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Warner      : int  0 0 0 1 0 0 0 0 1 1 ...
## $ Comp_20fox       : int  1 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Columbia    : int  0 0 1 0 0 1 0 0 0 0 ...
## $ Comp_NewLine     : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Disney      : int  0 1 0 0 1 0 1 0 0 0 ...
## $ Comp_VillageRoadshow: int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Miramax     : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Pixar       : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_RelMedia    : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_MGM         : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_DreamWorks  : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_Touchstone  : int  0 0 0 0 0 0 0 0 0 0 ...
## $ Comp_CanalPlus   : int  0 0 0 0 0 0 0 0 0 0 ...
## $ genere           : Factor w/ 19 levels "Action","Adventure",...: 9 2 1 1 1 1 3 1 9 1 ...
```

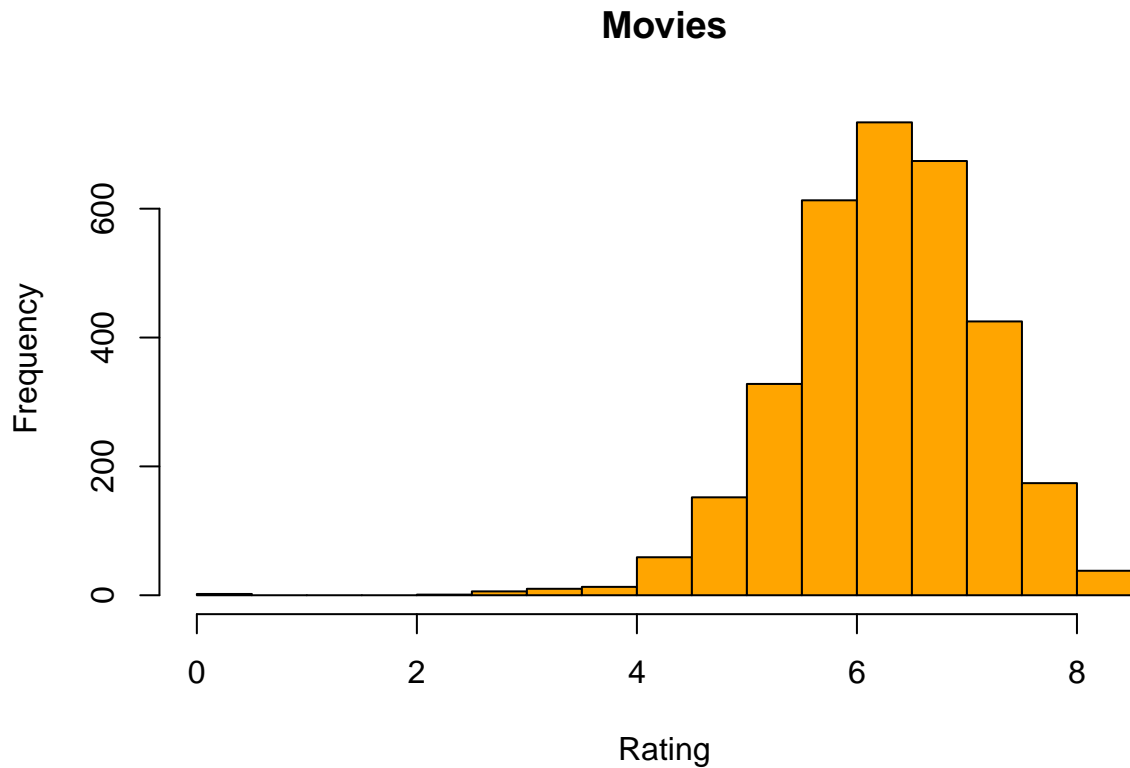
```
# Response variable: vote_average
summary(dati$vote_average)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      0.000   5.800   6.300   6.309   6.900   8.500
```

```
boxplot(dati$vote_average, col="orange", ylim=c(0,10), main="Movies", ylab="Rating")
```



```
hist(dati$vote_average, col="orange", main="Movies", xlab="Rating")
```



```
#Explanatory variables
summary(dati)
```

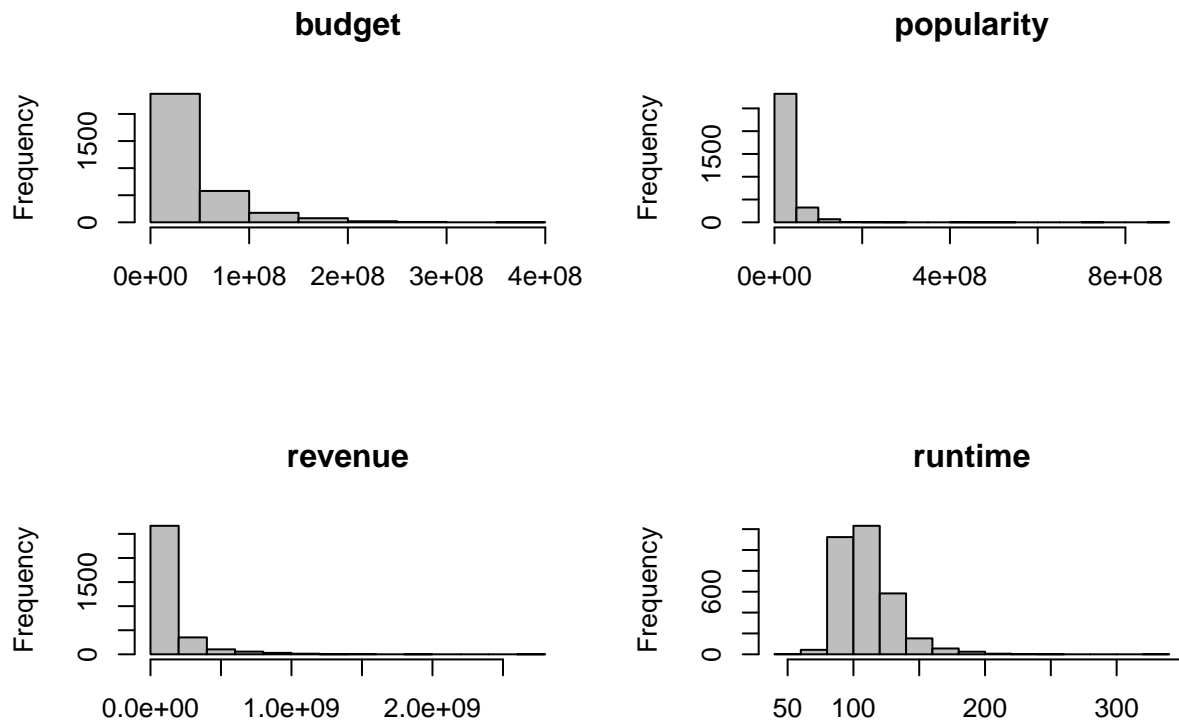
```
##      budget      popularity  production_countries
##  Min.   :      1  Min.   :      0  Min.   :0.0000
##  1st Qu.:10500000  1st Qu.: 7447714  1st Qu.:1.0000
##  Median :25000000  Median :18256028  Median :1.0000
##  Mean   :40654445  Mean   :26572600  Mean   :0.9006
##  3rd Qu.:55000000  3rd Qu.:35307577  3rd Qu.:1.0000
##  Max.   :380000000  Max.   :875581305  Max.   :1.0000
##
##  release_date      revenue      runtime  spoken_languages
##  Min.   :1916-09-04  Min.   :5.000e+00  Min.   : 41.0  Min.   :0.0000
##  1st Qu.:1998-09-11  1st Qu.:1.700e+07  1st Qu.: 96.0  1st Qu.:1.0000
##  Median :2005-07-20  Median :5.518e+07  Median :107.0  Median :1.0000
##  Mean   :2002-03-21  Mean   :1.212e+08  Mean   :110.7  Mean   :0.9709
##  3rd Qu.:2010-11-12  3rd Qu.:1.463e+08  3rd Qu.:121.0  3rd Qu.:1.0000
##  Max.   :2016-09-09  Max.   :2.788e+09  Max.   :338.0  Max.   :1.0000
##
##  vote_average  vote_classes  Comp_Universal  Comp_Paramount
##  Min.   :0.000  0 a 5 : 192  Min.   :0.00000  Min.   :0.00000
##  1st Qu.:5.800  5 a 6 : 843  1st Qu.:0.00000  1st Qu.:0.00000
##  Median :6.300  6 a 7 :1426  Median :0.00000  Median :0.00000
##  Mean   :6.309  7 a 8 : 706  Mean   :0.08826  Mean   :0.08083
##  3rd Qu.:6.900  8 a 10:  62  3rd Qu.:0.00000  3rd Qu.:0.00000
##  Max.   :8.500                Max.   :1.00000  Max.   :1.00000
```

```
##
##   Comp_Warner      Comp_20fox      Comp_Columbia      Comp_NewLine
##   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##   Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##   Mean   :0.09291   Mean   :0.06473   Mean   :0.08176   Mean   :0.04429
##   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##
##   Comp_Disney      Comp_VillageRoadshow  Comp_Miramax      Comp_Pixar
##   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##   Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##   Mean   :0.03685   Mean   :0.02261   Mean   :0.02323   Mean   :0.004955
##   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##
##   Comp_RelMedia      Comp_MGM      Comp_DreamWorks      Comp_Touchstone
##   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.0000
##   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000
##   Median :0.00000   Median :0.0000   Median :0.00000   Median :0.0000
##   Mean   :0.03097   Mean   :0.0288   Mean   :0.03066   Mean   :0.0288
##   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000
##   Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.0000
##
##   Comp_CanalPlus      genere
##   Min.   :0.00000   Action   :557
##   1st Qu.:0.00000   Drama    :556
##   Median :0.00000   Comedy   :487
##   Mean   :0.03345   Thriller :253
##   3rd Qu.:0.00000   Horror    :195
##   Max.   :1.00000   Animation:187
##                               (Other) :994
```

```
#We consider the plots of a subset of variables
summary(dati[,c(1,2,5,6)])
```

```
##      budget      popularity      revenue      runtime
##   Min.   :      1   Min.   :      0   Min.   :5.000e+00   Min.   : 41.0
##   1st Qu.:10500000   1st Qu.: 7447714   1st Qu.:1.700e+07   1st Qu.: 96.0
##   Median :25000000   Median :18256028   Median :5.518e+07   Median :107.0
##   Mean   :40654445   Mean   :26572600   Mean   :1.212e+08   Mean   :110.7
##   3rd Qu.:55000000   3rd Qu.:35307577   3rd Qu.:1.463e+08   3rd Qu.:121.0
##   Max.   :380000000   Max.   :875581305   Max.   :2.788e+09   Max.   :338.0
```

```
par(mfrow=c(2,2))
#We can see a right skewed behavior for three of them, so we need a log transformation
for(i in c(1,2,5,6)){
  hist(dati[,i], col="grey", main=paste(colnames(dati)[i]), xlab="")
}
```

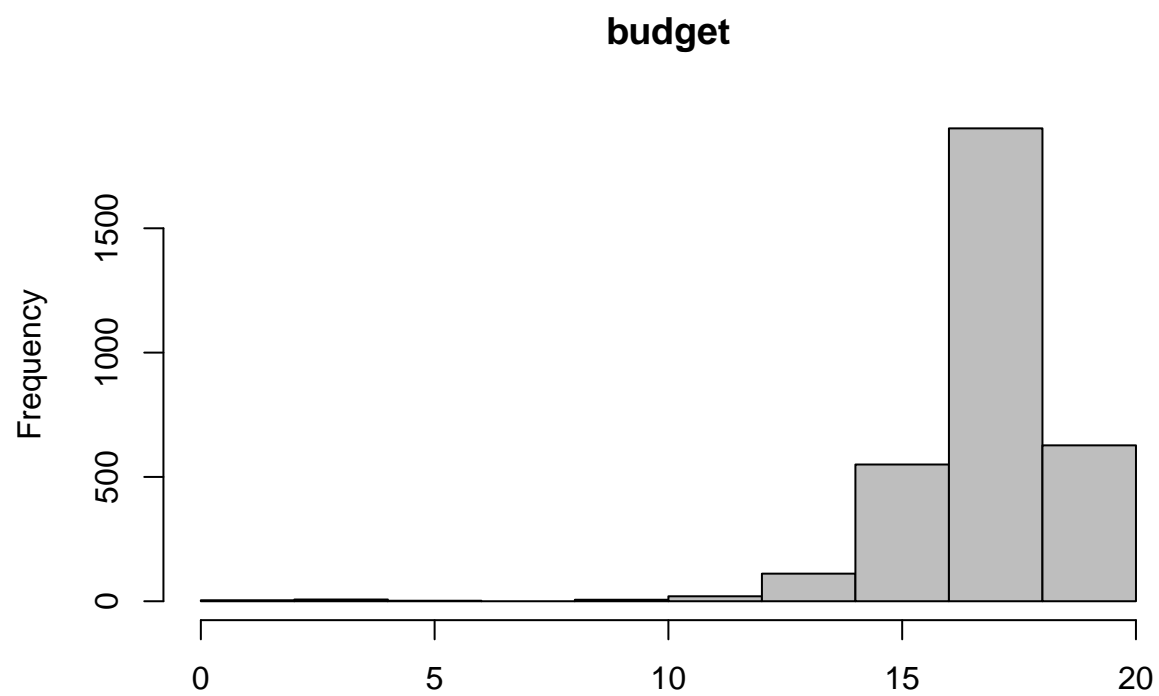


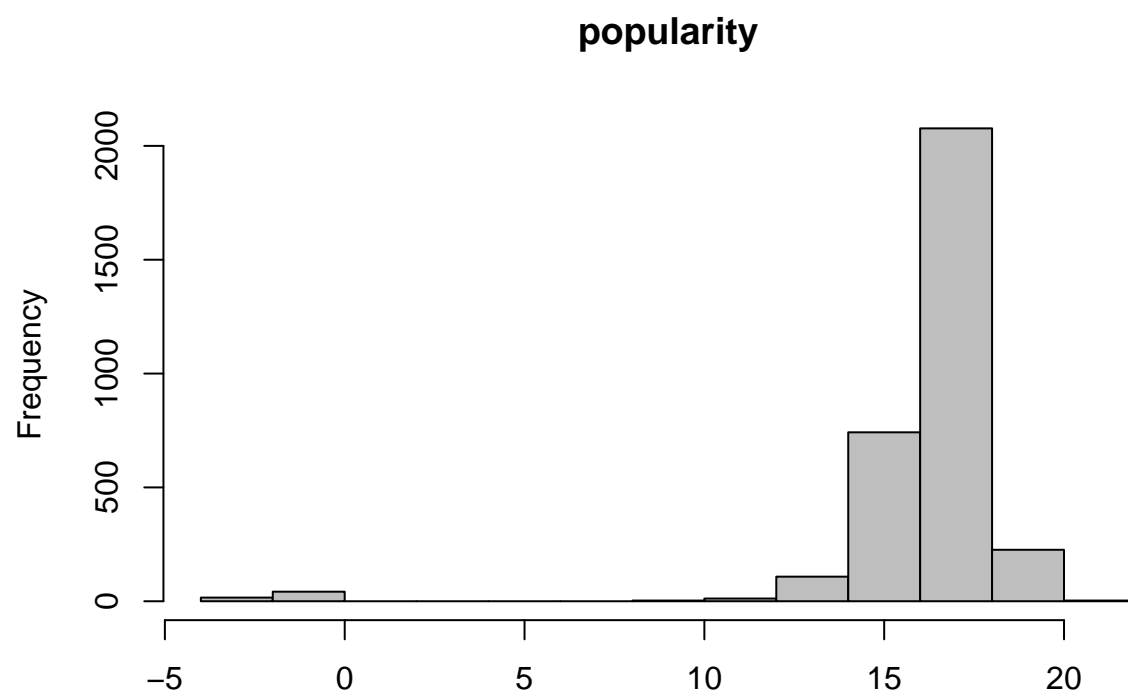
#Transform quantitative variables in log scale

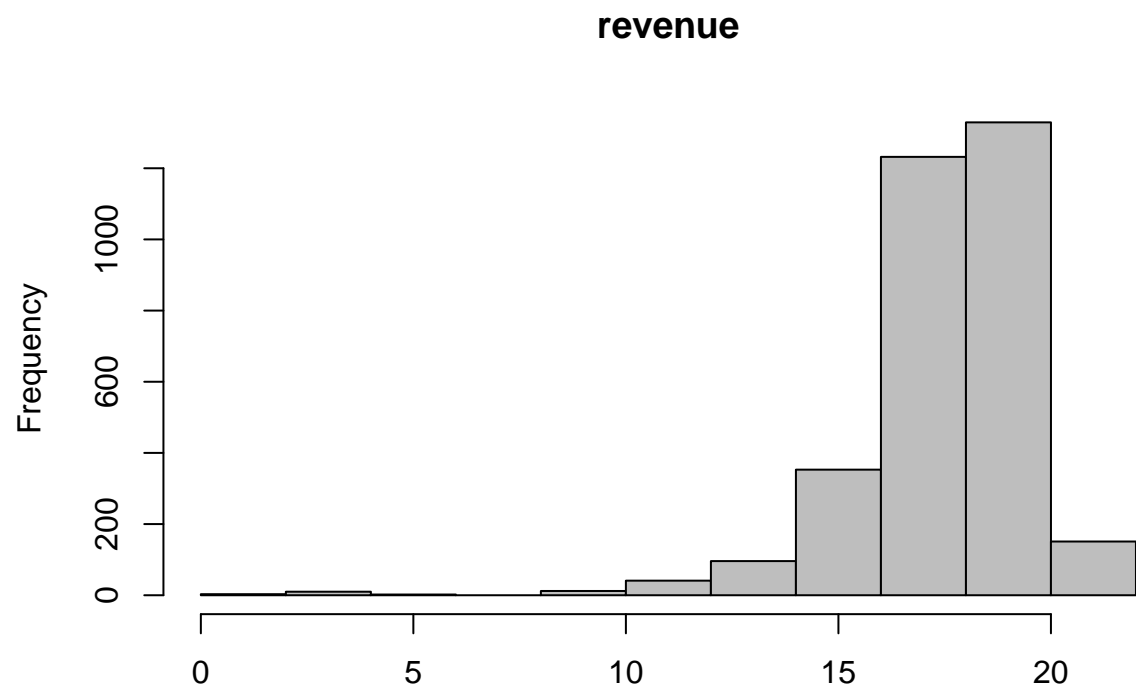
```
dati$budget <- log(dati$budget)
dati$popularity <- log(dati$popularity)
dati$revenue <- log(dati$revenue)
summary(dati[,c(1,2,5,6)])
```

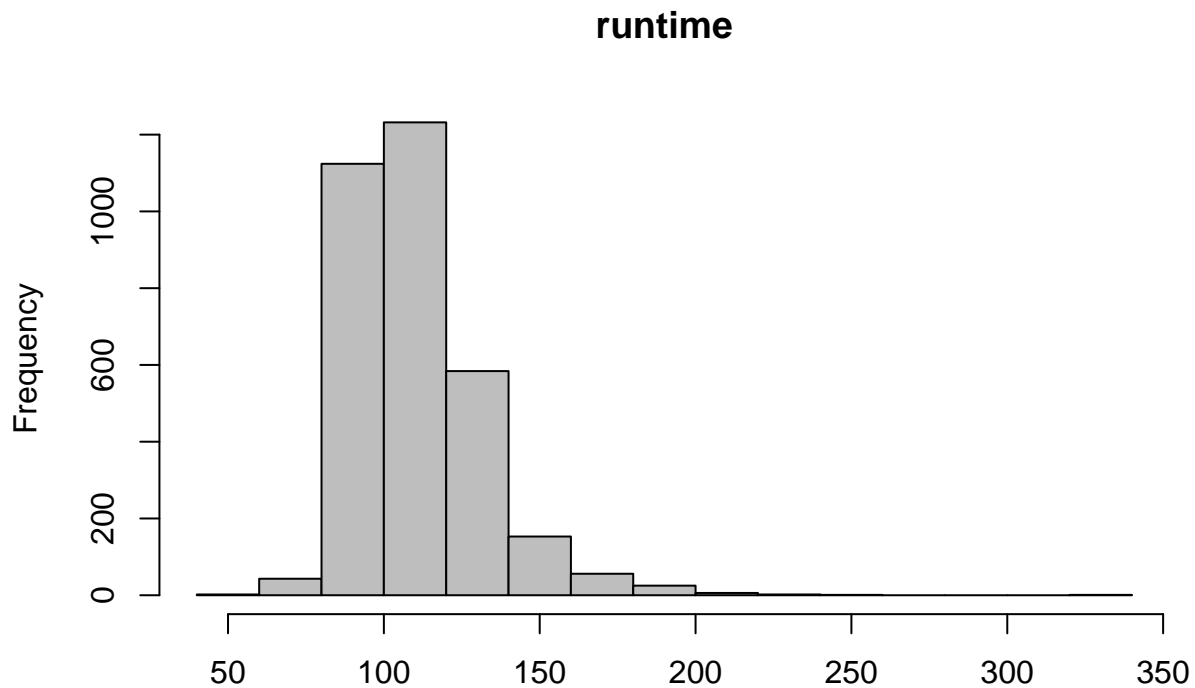
```
##      budget      popularity      revenue      runtime
## Min.   : 0.00   Min.   : -3.913   Min.   : 1.609   Min.   : 41.0
## 1st Qu.:16.17   1st Qu.:15.823   1st Qu.:16.649   1st Qu.: 96.0
## Median :17.03   Median :16.720   Median :17.826   Median :107.0
## Mean   :16.80   Mean   :16.213   Mean   :17.491   Mean   :110.7
## 3rd Qu.:17.82   3rd Qu.:17.380   3rd Qu.:18.801   3rd Qu.:121.0
## Max.   :19.76   Max.   :20.590   Max.   :21.749   Max.   :338.0
```

```
for(i in c(1,2,5,6)){
  hist(dati[,i], col="grey", main=paste(colnames(dati)[i]), xlab="")
}
```









```
par(mfrow=c(1,1))
```

```
#Transform release_date in numeric (to use gbm)
dati$release_date<-as.numeric(dati$release_date)
summary(dati$release_date)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -19477  10480   12984   11767   14925   17053
```

```
#Set train (70%) and test (30%)
set.seed(1)
train = sample (1:nrow(dati), 0.7*nrow(dati))
dati.train=dati[train ,]
dati.test=dati[-train ,]
```

```
#Make some variables factors
dati.train[,c(3,7, 10:24)]= lapply(dati.train[,c(3,7, 10:24)],factor)
dati.test[,c(3,7, 10:24)]= lapply(dati.test[,c(3,7, 10:24)],factor)
str(dati.train)
```

```
## 'data.frame':   2260 obs. of  25 variables:
## $ budget           : num  17.6 17.9 16.5 17.7 17.1 ...
## $ popularity       : num  15.1 16.8 16.7 15 16.6 ...
## $ production_countries: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 2 ...
```

```
## $ release_date      : num  9486 13138 16912 10172 13719 ...
## $ revenue           : num  16.4 18.7 16.8 16.2 18.3 ...
## $ runtime           : int   192 94 94 114 104 106 89 104 93 105 ...
## $ spoken_languages  : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 ...
## $ vote_average      : num   7.1 5.7 6 5.9 6.1 5.7 5.5 3.7 4.3 5.4 ...
## $ vote_classes      : Factor w/ 5 levels "0 a 5","5 a 6",...: 4 2 3 2 3 2 2 1 1 2 ...
## $ Comp_Universal    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_Paramount    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_Warner       : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 1 1 1 1 ...
## $ Comp_20fox        : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1 ...
## $ Comp_Columbia     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_NewLine      : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 2 1 2 ...
## $ Comp_Disney       : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_VillageRoadshow: Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
## $ Comp_Miramax      : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_Pixar        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_RelMedia     : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_MGM          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_DreamWorks   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_Touchstone   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ Comp_CanalPlus    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
## $ genere            : Factor w/ 19 levels "Action","Adventure",...: 10 4 1 17 4 9 11 4 4 4 ...
```

LINEAR MODEL

```
#We exclude the "class version" of our response variable
m1 <- lm(vote_average~.-vote_classes, data=dati.train)
summary(m1)
```

```
##
## Call:
## lm(formula = vote_average ~ . - vote_classes, data = dati.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.4049 -0.3962  0.0386  0.4333  4.2300
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.034e+00  2.030e-01  24.803 < 2e-16 ***
## budget        -1.897e-01  1.405e-02 -13.501 < 2e-16 ***
## popularity      8.045e-02  6.638e-03  12.120 < 2e-16 ***
## production_countries1 -2.428e-01  5.791e-02  -4.193 2.86e-05 ***
## release_date   -1.901e-05  3.744e-06  -5.078 4.13e-07 ***
## revenue        1.277e-01  1.071e-02  11.922 < 2e-16 ***
## runtime        1.278e-02  8.352e-04  15.304 < 2e-16 ***
## spoken_languages1 -2.528e-01  9.841e-02  -2.568 0.010279 *
## Comp_Universal1 -1.881e-02  5.525e-02  -0.340 0.733572
## Comp_Paramount1 -1.017e-01  5.660e-02  -1.797 0.072420 .
## Comp_Warner1    4.663e-02  5.564e-02   0.838 0.402033
## Comp_20fox1     -7.203e-02  6.187e-02  -1.164 0.244443
## Comp_Columbia1  -6.531e-02  5.652e-02  -1.155 0.248011
```

```
## Comp_NewLine1      -3.597e-02  7.147e-02  -0.503  0.614789
## Comp_Disney1       5.871e-02  8.406e-02   0.698  0.485001
## Comp_VillageRoadshow1 -7.348e-02  1.013e-01  -0.725  0.468319
## Comp_Miramax1      1.341e-01  9.710e-02   1.381  0.167469
## Comp_Pixar1        5.705e-01  2.223e-01   2.566  0.010352 *
## Comp_RelMedia1     -2.337e-02  8.906e-02  -0.262  0.793031
## Comp_MGM1          -5.353e-02  8.852e-02  -0.605  0.545433
## Comp_DreamWorks1   2.302e-01  8.691e-02   2.649  0.008131 **
## Comp_Touchstone1   7.122e-03  9.294e-02   0.077  0.938931
## Comp_CanalPlus1    5.012e-02  8.730e-02   0.574  0.565932
## genereAdventure     7.508e-02  9.187e-02   0.817  0.413872
## genereAnimation     4.862e-01  7.667e-02   6.341  2.75e-10 ***
## genereComedy        1.224e-02  5.219e-02   0.235  0.814573
## genereCrime         5.608e-01  8.551e-02   6.559  6.73e-11 ***
## genereDocumentary   9.107e-01  1.668e-01   5.459  5.31e-08 ***
## genereDrama         4.725e-01  5.138e-02   9.198  < 2e-16 ***
## genereFamily       -9.538e-04  1.675e-01  -0.006  0.995457
## genereFantasy       8.912e-02  7.653e-02   1.165  0.244335
## genereHistory       3.393e-01  1.046e-01   3.243  0.001199 **
## genereHorror        -1.402e-01  7.038e-02  -1.992  0.046506 *
## genereMystery       2.869e-01  3.468e-01   0.827  0.408129
## genereMusic         5.415e-01  1.175e-01   4.606  4.33e-06 ***
## genereMystery       3.461e-01  1.268e-01   2.729  0.006395 **
## genereRomance       3.006e-01  7.901e-02   3.805  0.000146 ***
## genereScience Fiction 3.344e-01  8.280e-02   4.038  5.57e-05 ***
## genereThriller      9.148e-02  6.464e-02   1.415  0.157184
## genereWar           5.847e-01  1.083e-01   5.400  7.38e-08 ***
## genereWestern       2.457e-01  1.362e-01   1.804  0.071386 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6884 on 2219 degrees of freedom
## Multiple R-squared:  0.3801, Adjusted R-squared:  0.369
## F-statistic: 34.02 on 40 and 2219 DF,  p-value: < 2.2e-16
```

Model selection with stepwise regression (the best one has the minimum AIC value. We can have also non-significant variables)

```
m2 <- step(m1, direction="both")
```

```
## Start:  AIC=-1646.84
## vote_average ~ (budget + popularity + production_countries +
##   release_date + revenue + runtime + spoken_languages + vote_classes +
##   Comp_Universal + Comp_Paramount + Comp_Warner + Comp_20fox +
##   Comp_Columbia + Comp_NewLine + Comp_Disney + Comp_VillageRoadshow +
##   Comp_Miramax + Comp_Pixar + Comp_RelMedia + Comp_MGM + Comp_DreamWorks +
##   Comp_Touchstone + Comp_CanalPlus + genere) - vote_classes
##
##
```

	Df	Sum of Sq	RSS	AIC
## - Comp_Touchstone	1	0.003	1051.7	-1648.8
## - Comp_RelMedia	1	0.033	1051.7	-1648.8
## - Comp_Universal	1	0.055	1051.7	-1648.7
## - Comp_NewLine	1	0.120	1051.8	-1648.6

```

## - Comp_CanalPlus      1      0.156 1051.8 -1648.5
## - Comp_MGM            1      0.173 1051.9 -1648.5
## - Comp_Disney         1      0.231 1051.9 -1648.3
## - Comp_VillageRoadshow 1      0.249 1051.9 -1648.3
## - Comp_Warner         1      0.333 1052.0 -1648.1
## - Comp_Columbia       1      0.633 1052.3 -1647.5
## - Comp_20fox          1      0.642 1052.3 -1647.5
## - Comp_Miramax        1      0.904 1052.6 -1646.9
## <none>                1051.7 -1646.8
## - Comp_Paramount      1      1.531 1053.2 -1645.5
## - Comp_Pixar          1      3.121 1054.8 -1642.1
## - spoken_languages     1      3.127 1054.8 -1642.1
## - Comp_DreamWorks     1      3.326 1055.0 -1641.7
## - production_countries 1      8.331 1060.0 -1631.0
## - release_date        1     12.222 1063.9 -1622.7
## - revenue             1     67.362 1119.0 -1508.5
## - popularity          1     69.618 1121.3 -1504.0
## - budget              1     86.383 1138.1 -1470.4
## - genere              18    112.663 1164.3 -1452.8
## - runtime             1    110.999 1162.7 -1422.1
##
## Step: AIC=-1648.83
## vote_average ~ budget + popularity + production_countries + release_date +
## revenue + runtime + spoken_languages + Comp_Universal + Comp_Paramount +
## Comp_Warner + Comp_20fox + Comp_Columbia + Comp_NewLine +
## Comp_Disney + Comp_VillageRoadshow + Comp_Miramax + Comp_Pixar +
## Comp_RelMedia + Comp_MGM + Comp_DreamWorks + Comp_CanalPlus +
## genere
##
##
##      Df Sum of Sq  RSS    AIC
## - Comp_RelMedia      1      0.033 1051.7 -1650.8
## - Comp_Universal      1      0.057 1051.7 -1650.7
## - Comp_NewLine        1      0.123 1051.8 -1650.6
## - Comp_CanalPlus      1      0.156 1051.8 -1650.5
## - Comp_MGM            1      0.176 1051.9 -1650.5
## - Comp_Disney         1      0.229 1051.9 -1650.3
## - Comp_VillageRoadshow 1      0.250 1051.9 -1650.3
## - Comp_Warner         1      0.330 1052.0 -1650.1
## - Comp_Columbia       1      0.644 1052.3 -1649.5
## - Comp_20fox          1      0.656 1052.3 -1649.4
## - Comp_Miramax        1      0.902 1052.6 -1648.9
## <none>                1051.7 -1648.8
## - Comp_Paramount      1      1.543 1053.2 -1647.5
## + Comp_Touchstone     1      0.003 1051.7 -1646.8
## - Comp_Pixar          1      3.121 1054.8 -1644.1
## - spoken_languages     1      3.126 1054.8 -1644.1
## - Comp_DreamWorks     1      3.336 1055.0 -1643.7
## - production_countries 1      8.340 1060.0 -1633.0
## - release_date        1     12.259 1063.9 -1624.6
## - revenue             1     67.385 1119.1 -1510.5
## - popularity          1     69.665 1121.3 -1505.9
## - budget              1     86.662 1138.3 -1471.9
## - genere              18    112.704 1164.4 -1454.8
## - runtime             1    111.076 1162.8 -1423.9

```

```
##
## Step: AIC=-1650.76
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Universal + Comp_Paramount +
##     Comp_Warner + Comp_20fox + Comp_Columbia + Comp_NewLine +
##     Comp_Disney + Comp_VillageRoadshow + Comp_Miramax + Comp_Pixar +
##     Comp_MGM + Comp_DreamWorks + Comp_CanalPlus + genere
##
##      Df Sum of Sq    RSS    AIC
## - Comp_Universal      1      0.079 1051.8 -1652.6
## - Comp_NewLine        1      0.119 1051.8 -1652.5
## - Comp_CanalPlus      1      0.152 1051.9 -1652.4
## - Comp_MGM            1      0.179 1051.9 -1652.4
## - Comp_Disney         1      0.233 1052.0 -1652.3
## - Comp_VillageRoadshow 1      0.246 1052.0 -1652.2
## - Comp_Warner         1      0.337 1052.0 -1652.0
## - Comp_20fox          1      0.649 1052.4 -1651.4
## - Comp_Columbia       1      0.687 1052.4 -1651.3
## - Comp_Miramax        1      0.913 1052.6 -1650.8
## <none>                  1051.7 -1650.8
## - Comp_Paramount      1      1.533 1053.2 -1649.5
## + Comp_RelMedia       1      0.033 1051.7 -1648.8
## + Comp_Touchstone     1      0.003 1051.7 -1648.8
## - Comp_Pixar          1      3.123 1054.8 -1646.1
## - spoken_languages     1      3.129 1054.8 -1646.0
## - Comp_DreamWorks     1      3.363 1055.1 -1645.5
## - production_countries 1      8.416 1060.1 -1634.8
## - release_date        1     12.535 1064.2 -1626.0
## - revenue             1     67.353 1119.1 -1512.5
## - popularity          1     69.693 1121.4 -1507.8
## - budget              1     86.638 1138.4 -1473.9
## - genere              18    112.682 1164.4 -1456.7
## - runtime             1    111.107 1162.8 -1425.8
##
## Step: AIC=-1652.59
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Paramount + Comp_Warner +
##     Comp_20fox + Comp_Columbia + Comp_NewLine + Comp_Disney +
##     Comp_VillageRoadshow + Comp_Miramax + Comp_Pixar + Comp_MGM +
##     Comp_DreamWorks + Comp_CanalPlus + genere
##
##      Df Sum of Sq    RSS    AIC
## - Comp_NewLine      1      0.101 1051.9 -1654.4
## - Comp_CanalPlus     1      0.136 1051.9 -1654.3
## - Comp_MGM           1      0.161 1052.0 -1654.2
## - Comp_VillageRoadshow 1      0.238 1052.0 -1654.1
## - Comp_Disney        1      0.274 1052.1 -1654.0
## - Comp_Warner        1      0.396 1052.2 -1653.7
## - Comp_20fox         1      0.594 1052.4 -1653.3
## - Comp_Columbia      1      0.636 1052.4 -1653.2
## <none>                1051.8 -1652.6
## - Comp_Miramax       1      0.932 1052.7 -1652.6
## - Comp_Paramount     1      1.469 1053.3 -1651.4
## + Comp_Universal     1      0.079 1051.7 -1650.8
```

```

## + Comp_RelMedia      1      0.056 1051.7 -1650.7
## + Comp_Touchstone    1      0.006 1051.8 -1650.6
## - Comp_Pixar          1      3.145 1054.9 -1647.8
## - spoken_languages    1      3.147 1055.0 -1647.8
## - Comp_DreamWorks     1      3.436 1055.2 -1647.2
## - production_countries 1      8.577 1060.4 -1636.2
## - release_date        1     12.457 1064.2 -1628.0
## - revenue             1     67.475 1119.3 -1514.1
## - popularity          1     69.721 1121.5 -1509.5
## - budget              1     86.851 1138.7 -1475.3
## - genere              18    113.428 1165.2 -1457.1
## - runtime             1    111.027 1162.8 -1427.8
##
## Step: AIC=-1654.38
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Paramount + Comp_Warner +
##     Comp_20fox + Comp_Columbia + Comp_Disney + Comp_VillageRoadshow +
##     Comp_Miramax + Comp_Pixar + Comp_MGM + Comp_DreamWorks +
##     Comp_CanalPlus + genere
##
##              Df Sum of Sq    RSS      AIC
## - Comp_CanalPlus      1      0.146 1052.0 -1656.1
## - Comp_MGM             1      0.162 1052.1 -1656.0
## - Comp_VillageRoadshow 1      0.236 1052.1 -1655.9
## - Comp_Disney          1      0.299 1052.2 -1655.7
## - Comp_Warner          1      0.427 1052.3 -1655.5
## - Comp_20fox           1      0.553 1052.5 -1655.2
## - Comp_Columbia        1      0.599 1052.5 -1655.1
## <none>                  1051.9 -1654.4
## - Comp_Miramax         1      0.954 1052.8 -1654.3
## - Comp_Paramount       1      1.410 1053.3 -1653.3
## + Comp_NewLine         1      0.101 1051.8 -1652.6
## + Comp_Universal       1      0.061 1051.8 -1652.5
## + Comp_RelMedia        1      0.048 1051.8 -1652.5
## + Comp_Touchstone      1      0.010 1051.9 -1652.4
## - Comp_Pixar           1      3.148 1055.0 -1649.6
## - spoken_languages     1      3.175 1055.1 -1649.6
## - Comp_DreamWorks      1      3.501 1055.4 -1648.9
## - production_countries 1      8.654 1060.5 -1637.9
## - release_date         1     12.475 1064.4 -1629.7
## - revenue              1     67.427 1119.3 -1516.0
## - popularity           1     69.811 1121.7 -1511.2
## - budget               1     86.812 1138.7 -1477.2
## - genere               18    114.220 1166.1 -1457.4
## - runtime              1    111.063 1163.0 -1429.5
##
## Step: AIC=-1656.06
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Paramount + Comp_Warner +
##     Comp_20fox + Comp_Columbia + Comp_Disney + Comp_VillageRoadshow +
##     Comp_Miramax + Comp_Pixar + Comp_MGM + Comp_DreamWorks +
##     genere
##
##              Df Sum of Sq    RSS      AIC

```

```

## - Comp_MGM 1 0.158 1052.2 -1657.7
## - Comp_VillageRoadshow 1 0.248 1052.3 -1657.5
## - Comp_Disney 1 0.290 1052.3 -1657.4
## - Comp_Warner 1 0.431 1052.5 -1657.1
## - Comp_20fox 1 0.547 1052.6 -1656.9
## - Comp_Columbia 1 0.632 1052.7 -1656.7
## <none> 1052.0 -1656.1
## - Comp_Miramax 1 0.957 1053.0 -1656.0
## - Comp_Paramount 1 1.447 1053.5 -1655.0
## + Comp_CanalPlus 1 0.146 1051.9 -1654.4
## + Comp_NewLine 1 0.110 1051.9 -1654.3
## + Comp_Universal 1 0.047 1052.0 -1654.2
## + Comp_RelMedia 1 0.040 1052.0 -1654.2
## + Comp_Touchstone 1 0.009 1052.0 -1654.1
## - Comp_Pixar 1 3.155 1055.2 -1651.3
## - spoken_languages 1 3.193 1055.2 -1651.2
## - Comp_DreamWorks 1 3.470 1055.5 -1650.6
## - production_countries 1 9.350 1061.4 -1638.1
## - release_date 1 12.407 1064.5 -1631.6
## - revenue 1 67.353 1119.4 -1517.8
## - popularity 1 70.057 1122.1 -1512.4
## - budget 1 86.667 1138.7 -1479.2
## - genere 18 114.126 1166.2 -1459.3
## - runtime 1 110.983 1163.0 -1431.4
##
## Step: AIC=-1657.72
## vote_average ~ budget + popularity + production_countries + release_date +
## revenue + runtime + spoken_languages + Comp_Paramount + Comp_Warner +
## Comp_20fox + Comp_Columbia + Comp_Disney + Comp_VillageRoadshow +
## Comp_Miramax + Comp_Pixar + Comp_DreamWorks + genere
##
## Df Sum of Sq RSS AIC
## - Comp_VillageRoadshow 1 0.241 1052.4 -1659.2
## - Comp_Disney 1 0.312 1052.5 -1659.0
## - Comp_Warner 1 0.462 1052.7 -1658.7
## - Comp_20fox 1 0.506 1052.7 -1658.6
## - Comp_Columbia 1 0.615 1052.8 -1658.4
## <none> 1052.2 -1657.7
## - Comp_Miramax 1 0.985 1053.2 -1657.6
## - Comp_Paramount 1 1.403 1053.6 -1656.7
## + Comp_MGM 1 0.158 1052.0 -1656.1
## + Comp_CanalPlus 1 0.141 1052.1 -1656.0
## + Comp_NewLine 1 0.112 1052.1 -1656.0
## + Comp_RelMedia 1 0.040 1052.2 -1655.8
## + Comp_Universal 1 0.034 1052.2 -1655.8
## + Comp_Touchstone 1 0.013 1052.2 -1655.8
## - Comp_Pixar 1 3.161 1055.4 -1652.9
## - spoken_languages 1 3.198 1055.4 -1652.9
## - Comp_DreamWorks 1 3.528 1055.7 -1652.2
## - production_countries 1 9.489 1061.7 -1639.4
## - release_date 1 12.255 1064.5 -1633.5
## - revenue 1 67.304 1119.5 -1519.6
## - popularity 1 70.515 1122.7 -1513.1
## - budget 1 87.381 1139.6 -1479.4

```

```

## - genere          18   114.029 1166.2 -1461.2
## - runtime         1    110.980 1163.2 -1433.1
##
## Step: AIC=-1659.21
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Paramount + Comp_Warner +
##     Comp_20fox + Comp_Columbia + Comp_Disney + Comp_Miramax +
##     Comp_Pixar + Comp_DreamWorks + genere
##
##              Df Sum of Sq    RSS    AIC
## - Comp_Warner      1      0.308 1052.8 -1660.5
## - Comp_Disney      1      0.324 1052.8 -1660.5
## - Comp_20fox       1      0.491 1052.9 -1660.2
## - Comp_Columbia    1      0.634 1053.1 -1659.8
## <none>              1052.4 -1659.2
## - Comp_Miramax     1      0.995 1053.4 -1659.1
## - Comp_Paramount   1      1.390 1053.8 -1658.2
## + Comp_VillageRoadshow 1      0.241 1052.2 -1657.7
## + Comp_CanalPlus   1      0.153 1052.3 -1657.5
## + Comp_MGM         1      0.151 1052.3 -1657.5
## + Comp_NewLine     1      0.110 1052.3 -1657.4
## + Comp_RelMedia    1      0.034 1052.4 -1657.3
## + Comp_Universal   1      0.028 1052.4 -1657.3
## + Comp_Touchstone  1      0.014 1052.4 -1657.2
## - Comp_Pixar       1      3.163 1055.6 -1654.4
## - spoken_languages  1      3.164 1055.6 -1654.4
## - Comp_DreamWorks  1      3.584 1056.0 -1653.5
## - production_countries 1      9.567 1062.0 -1640.8
## - release_date     1     12.419 1064.9 -1634.7
## - revenue          1     67.209 1119.7 -1521.3
## - popularity       1     70.676 1123.1 -1514.3
## - budget           1     87.797 1140.2 -1480.1
## - genere          18    114.348 1166.8 -1462.1
## - runtime         1    111.912 1164.4 -1432.8
##
## Step: AIC=-1660.54
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Paramount + Comp_20fox +
##     Comp_Columbia + Comp_Disney + Comp_Miramax + Comp_Pixar +
##     Comp_DreamWorks + genere
##
##              Df Sum of Sq    RSS    AIC
## - Comp_Disney      1      0.270 1053.0 -1662.0
## - Comp_20fox       1      0.604 1053.4 -1661.2
## - Comp_Columbia    1      0.777 1053.5 -1660.9
## <none>              1052.8 -1660.5
## - Comp_Miramax     1      0.942 1053.7 -1660.5
## + Comp_Warner      1      0.308 1052.4 -1659.2
## - Comp_Paramount   1      1.580 1054.3 -1659.2
## + Comp_MGM         1      0.179 1052.6 -1658.9
## + Comp_CanalPlus   1      0.152 1052.6 -1658.9
## + Comp_NewLine     1      0.140 1052.6 -1658.8
## + Comp_VillageRoadshow 1      0.087 1052.7 -1658.7
## + Comp_Universal   1      0.063 1052.7 -1658.7

```



```

## + Comp_RelMedia      1      0.049 1052.7 -1658.7
## + Comp_Touchstone    1      0.005 1052.8 -1658.6
## - Comp_Pixar         1      3.117 1055.9 -1655.9
## - spoken_languages    1      3.160 1055.9 -1655.8
## - Comp_DreamWorks     1      3.496 1056.2 -1655.0
## - production_countries 1      9.370 1062.1 -1642.5
## - release_date        1     12.850 1065.6 -1635.1
## - revenue             1     67.163 1119.9 -1522.8
## - popularity          1     70.963 1123.7 -1515.1
## - budget              1     87.637 1140.4 -1481.8
## - genere              18    114.225 1167.0 -1463.8
## - runtime             1    113.620 1166.4 -1430.9
##
## Step: AIC=-1661.97
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Paramount + Comp_20fox +
##     Comp_Columbia + Comp_Miramax + Comp_Pixar + Comp_DreamWorks +
##     genere
##
##              Df Sum of Sq  RSS    AIC
## - Comp_20fox      1      0.688 1053.7 -1662.5
## - Comp_Columbia    1      0.872 1053.9 -1662.1
## - Comp_Miramax     1      0.913 1053.9 -1662.0
## <none>              1053.0 -1662.0
## + Comp_Disney      1      0.270 1052.8 -1660.5
## + Comp_Warner       1      0.253 1052.8 -1660.5
## + Comp_MGM          1      0.199 1052.8 -1660.4
## - Comp_Paramount    1      1.678 1054.7 -1660.4
## + Comp_NewLine      1      0.164 1052.9 -1660.3
## + Comp_CanalPlus    1      0.143 1052.9 -1660.3
## + Comp_VillageRoadshow 1      0.103 1052.9 -1660.2
## + Comp_Universal    1      0.092 1052.9 -1660.2
## + Comp_RelMedia     1      0.059 1053.0 -1660.1
## + Comp_Touchstone   1      0.003 1053.0 -1660.0
## - spoken_languages    1      3.162 1056.2 -1657.2
## - Comp_DreamWorks     1      3.349 1056.4 -1656.8
## - Comp_Pixar         1      3.593 1056.6 -1656.3
## - production_countries 1      9.255 1062.3 -1644.2
## - release_date        1     12.920 1065.9 -1636.4
## - revenue             1     67.757 1120.8 -1523.0
## - popularity          1     70.894 1123.9 -1516.7
## - budget              1     87.406 1140.4 -1483.8
## - genere              18    114.594 1167.6 -1464.5
## - runtime             1    113.523 1166.5 -1432.6
##
## Step: AIC=-1662.49
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Paramount + Comp_Columbia +
##     Comp_Miramax + Comp_Pixar + Comp_DreamWorks + genere
##
##              Df Sum of Sq  RSS    AIC
## - Comp_Columbia      1      0.720 1054.4 -1663.0
## <none>              1053.7 -1662.5
## - Comp_Miramax       1      0.961 1054.7 -1662.4

```

```

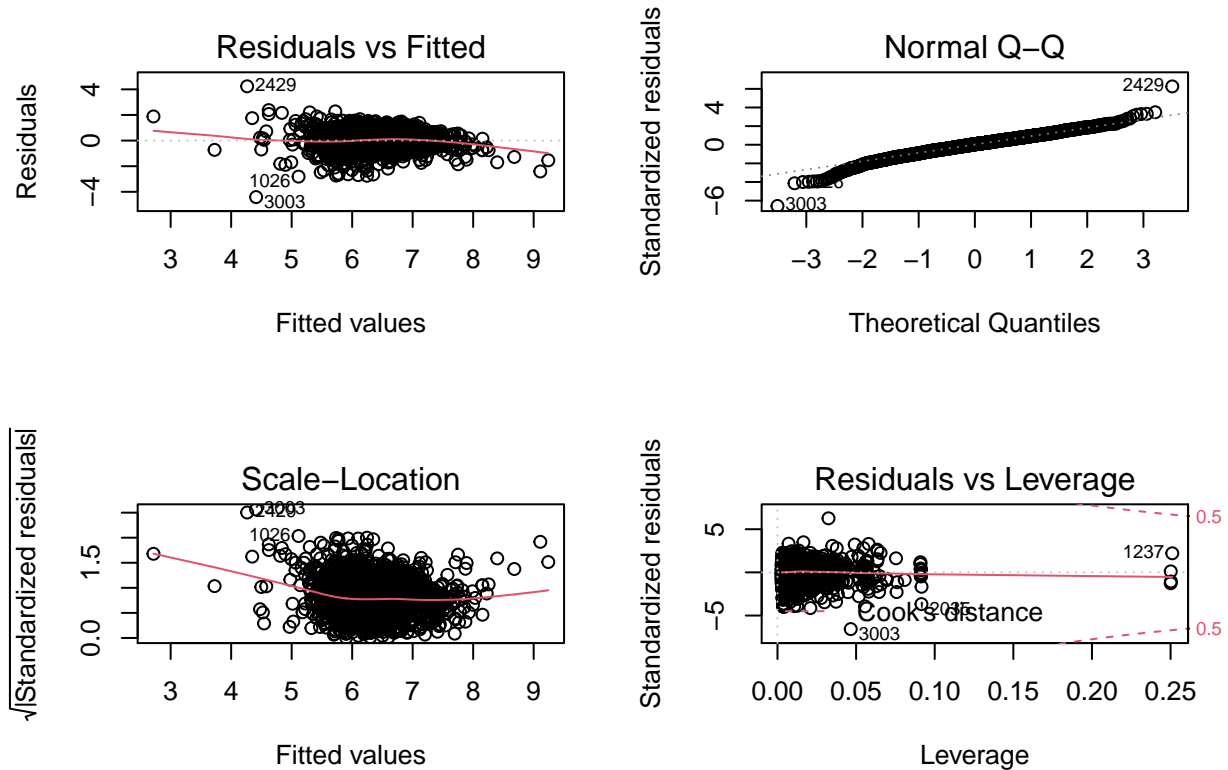
## + Comp_20fox          1      0.688 1053.0 -1662.0
## - Comp_Paramount      1      1.480 1055.2 -1661.3
## + Comp_Warner         1      0.358 1053.3 -1661.3
## + Comp_Disney         1      0.353 1053.4 -1661.2
## + Comp_MGM            1      0.152 1053.6 -1660.8
## + Comp_CanalPlus      1      0.135 1053.6 -1660.8
## + Comp_NewLine        1      0.113 1053.6 -1660.7
## + Comp_VillageRoadshow 1      0.075 1053.6 -1660.7
## + Comp_RelMedia       1      0.044 1053.7 -1660.6
## + Comp_Universal      1      0.044 1053.7 -1660.6
## + Comp_Touchstone     1      0.010 1053.7 -1660.5
## - spoken_languages    1      3.190 1056.9 -1657.7
## - Comp_DreamWorks     1      3.322 1057.0 -1657.4
## - Comp_Pixar          1      3.729 1057.4 -1656.5
## - production_countries 1      9.367 1063.1 -1644.5
## - release_date        1     12.429 1066.1 -1638.0
## - revenue             1     67.127 1120.8 -1524.9
## - popularity          1     70.768 1124.5 -1517.6
## - budget              1     87.929 1141.6 -1483.4
## - genere              18    115.412 1169.1 -1463.6
## - runtime             1    113.896 1167.6 -1432.5
##
## Step: AIC=-1662.95
## vote_average ~ budget + popularity + production_countries + release_date +
##     revenue + runtime + spoken_languages + Comp_Paramount + Comp_Miramax +
##     Comp_Pixar + Comp_DreamWorks + genere
##
##              Df Sum of Sq    RSS      AIC
## <none>                1054.4 -1663.0
## - Comp_Miramax        1      1.053 1055.5 -1662.7
## + Comp_Columbia       1      0.720 1053.7 -1662.5
## - Comp_Paramount      1      1.294 1055.7 -1662.2
## + Comp_20fox          1      0.537 1053.9 -1662.1
## + Comp_Warner         1      0.481 1054.0 -1662.0
## + Comp_Disney         1      0.439 1054.0 -1661.9
## + Comp_CanalPlus      1      0.171 1054.3 -1661.3
## + Comp_MGM            1      0.146 1054.3 -1661.3
## + Comp_RelMedia       1      0.088 1054.3 -1661.1
## + Comp_NewLine        1      0.082 1054.3 -1661.1
## + Comp_VillageRoadshow 1      0.070 1054.4 -1661.1
## + Comp_Touchstone     1      0.019 1054.4 -1661.0
## + Comp_Universal      1      0.017 1054.4 -1661.0
## - spoken_languages    1      3.156 1057.6 -1658.2
## - Comp_DreamWorks     1      3.495 1057.9 -1657.5
## - Comp_Pixar          1      3.854 1058.3 -1656.7
## - production_countries 1      9.516 1064.0 -1644.6
## - release_date        1     12.021 1066.5 -1639.3
## - revenue             1     66.782 1121.2 -1526.2
## - popularity          1     70.303 1124.7 -1519.1
## - budget              1     89.408 1143.8 -1481.0
## - genere              18    114.879 1169.3 -1465.2
## - runtime             1    114.287 1168.7 -1432.4

```

```
summary(m2)
```

```
##
## Call:
## lm(formula = vote_average ~ budget + popularity + production_countries +
##     release_date + revenue + runtime + spoken_languages + Comp_Paramount +
##     Comp_Miramax + Comp_Pixar + Comp_DreamWorks + genere, data = dati.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.4125 -0.3976  0.0353  0.4312  4.2355
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.055e+00  1.984e-01  25.473 < 2e-16 ***
## budget        -1.903e-01  1.384e-02 -13.751 < 2e-16 ***
## popularity      8.055e-02  6.606e-03  12.194 < 2e-16 ***
## production_countries1 -2.525e-01  5.628e-02  -4.486 7.62e-06 ***
## release_date   -1.826e-05  3.621e-06  -5.042 4.97e-07 ***
## revenue         1.257e-01  1.058e-02  11.884 < 2e-16 ***
## runtime         1.289e-02  8.293e-04  15.547 < 2e-16 ***
## spoken_languages1 -2.537e-01  9.821e-02  -2.584 0.009842 **
## Comp_Paramount1  -9.013e-02  5.448e-02  -1.654 0.098219 .
## Comp_Miramax1    1.439e-01  9.641e-02   1.492 0.135724
## Comp_Pixar1      6.217e-01  2.177e-01   2.855 0.004343 **
## Comp_DreamWorks1 2.332e-01  8.578e-02   2.719 0.006604 **
## genereAdventure   8.185e-02  9.132e-02   0.896 0.370210
## genereAnimation   4.950e-01  7.514e-02   6.588 5.56e-11 ***
## genereComedy       7.288e-03  5.176e-02   0.141 0.888043
## genereCrime        5.569e-01  8.518e-02   6.538 7.72e-11 ***
## genereDocumentary 9.098e-01  1.664e-01   5.466 5.11e-08 ***
## genereDrama        4.723e-01  5.106e-02   9.249 < 2e-16 ***
## genereFamily       6.213e-03  1.664e-01   0.037 0.970209
## genereFantasy      9.442e-02  7.620e-02   1.239 0.215485
## genereHistory      3.444e-01  1.044e-01   3.299 0.000984 ***
## genereHorror      -1.424e-01  7.000e-02  -2.034 0.042022 *
## genereMystery      2.552e-01  3.457e-01   0.738 0.460508
## genereMusic        5.359e-01  1.170e-01   4.581 4.87e-06 ***
## genereMystery      3.517e-01  1.263e-01   2.784 0.005418 **
## genereRomance      2.979e-01  7.863e-02   3.788 0.000156 ***
## genereScience Fiction 3.296e-01  8.247e-02   3.996 6.64e-05 ***
## genereThriller     9.110e-02  6.441e-02   1.414 0.157404
## genereWar          5.860e-01  1.078e-01   5.436 6.03e-08 ***
## genereWestern      2.584e-01  1.357e-01   1.904 0.057038 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6876 on 2230 degrees of freedom
## Multiple R-squared:  0.3785, Adjusted R-squared:  0.3704
## F-statistic: 46.83 on 29 and 2230 DF, p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(m2)
```



```
par(mfrow=c(1,1))
```

Considerations: “budget” has a negative role respect our response, we have to consider that the votes are given by committees and not normal people, so they could consider in a positive way films with a low budget that in most cases are independent film. Another consideration is that “revenue” a strong positive one

```
#Prediction
p.lm <- predict(m2, newdata=dati.test)
#This is the deviance of our selected model. This value can be compared with the deviance of other
#model (see later GAM model)
dev.lm <- sum((p.lm-dati.test$vote_average)^2)
dev.lm
```

```
## [1] 512.1419
```

```
AIC(m2)
```

```
## [1] 4752.656
```

GAM

GAM with splines. Considering that we want use some variables with smoothing splines, and this is a problem with factorial variables, we create a formula to recognize variables that allow the use of splines and variables that don't allow it. We will use splines on all the var. that allow it (only numericals, not factors)

```
library(gam)
fg1 <- as.formula(paste(paste("dati.train[,8] ~s(", paste(names(dati.train[c(1,2,4,5,6)]),collapse=") +
fg1
```

```
## dati.train[, 8] ~ s(budget) + s(popularity) + s(release_date) +
##      s(revenue) + s(runtime) + production_countries + spoken_languages +
##      Comp_Universal + Comp_Paramount + Comp_Warner + Comp_20fox +
##      Comp_Columbia + Comp_NewLine + Comp_Disney + Comp_VillageRoadshow +
##      Comp_Miramax + Comp_Pixar + Comp_RelMedia + Comp_MGM + Comp_DreamWorks +
##      Comp_Touchstone + Comp_CanalPlus + genere
```

#Here we don't specify any df, it uses the default value 4: for each var.with smoothing 1 for the par. part and 3 for the non-par. part (see summary)

```
g1<-gam(fg1,data=dati.train)
summary(g1)
```

```
##
## Call: gam(formula = fg1, data = dati.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -5.24386 -0.35939  0.02371  0.39858  3.50692
##
## (Dispersion Parameter for gaussian family taken to be 0.4094)
##
##      Null Deviance: 1696.602 on 2259 degrees of freedom
## Residual Deviance: 902.3782 on 2204 degrees of freedom
## AIC: 4452.726
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##


|                      | Df | Sum Sq | Mean Sq | F value  | Pr(>F)        |
|----------------------|----|--------|---------|----------|---------------|
| s(budget)            | 1  | 60.51  | 60.513  | 147.7982 | < 2.2e-16 *** |
| s(popularity)        | 1  | 147.13 | 147.131 | 359.3568 | < 2.2e-16 *** |
| s(release_date)      | 1  | 55.07  | 55.067  | 134.4969 | < 2.2e-16 *** |
| s(revenue)           | 1  | 35.47  | 35.472  | 86.6379  | < 2.2e-16 *** |
| s(runtime)           | 1  | 201.82 | 201.824 | 492.9424 | < 2.2e-16 *** |
| production_countries | 1  | 16.53  | 16.525  | 40.3621  | 2.555e-10 *** |
| spoken_languages     | 1  | 1.48   | 1.476   | 3.6049   | 0.0577417 .   |
| Comp_Universal       | 1  | 0.43   | 0.434   | 1.0600   | 0.3033360     |
| Comp_Paramount       | 1  | 0.93   | 0.925   | 2.2596   | 0.1329307     |
| Comp_Warner          | 1  | 0.50   | 0.502   | 1.2262   | 0.2682647     |
| Comp_20fox           | 1  | 1.07   | 1.070   | 2.6142   | 0.1060557     |
| Comp_Columbia        | 1  | 0.28   | 0.281   | 0.6865   | 0.4074469     |
| Comp_NewLine         | 1  | 1.23   | 1.230   | 3.0035   | 0.0832230 .   |
| Comp_Disney          | 1  | 3.03   | 3.025   | 7.3885   | 0.0066158 **  |
| Comp_VillageRoadshow | 1  | 0.10   | 0.098   | 0.2382   | 0.6255703     |

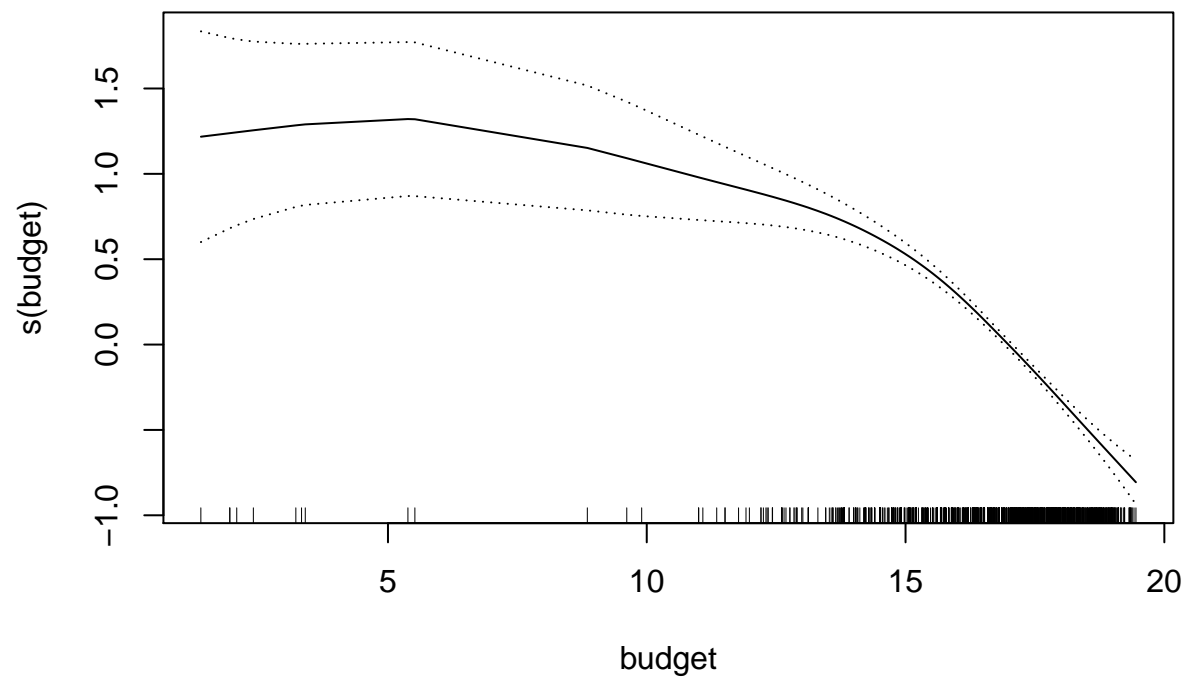

```

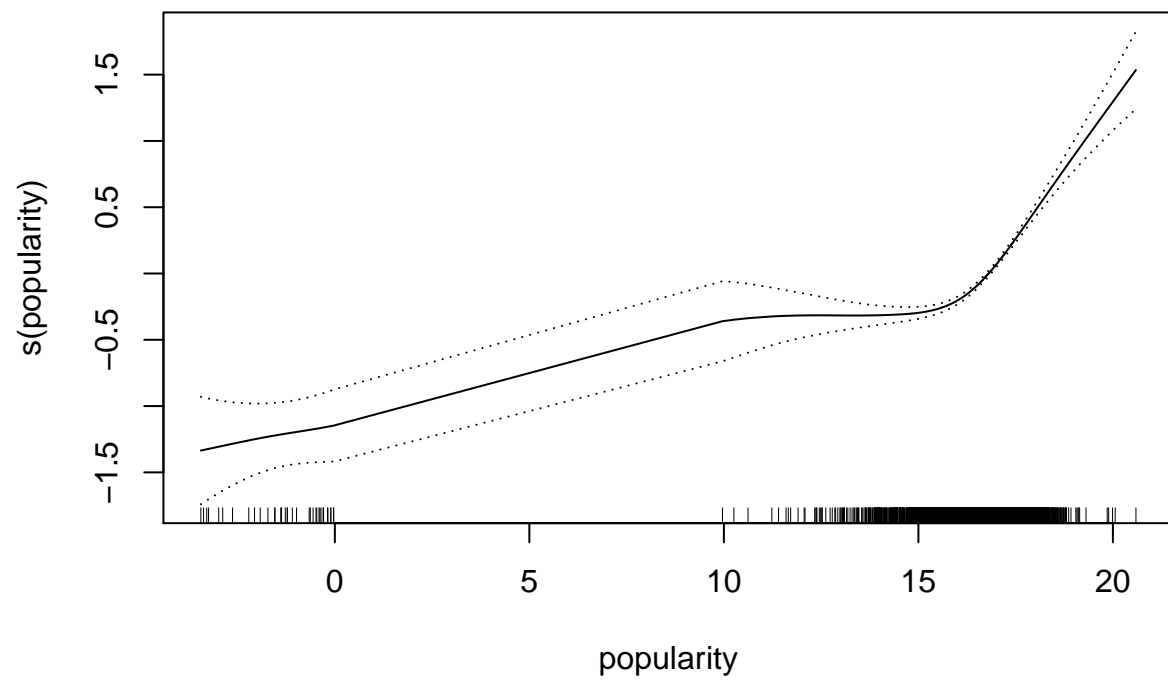
```

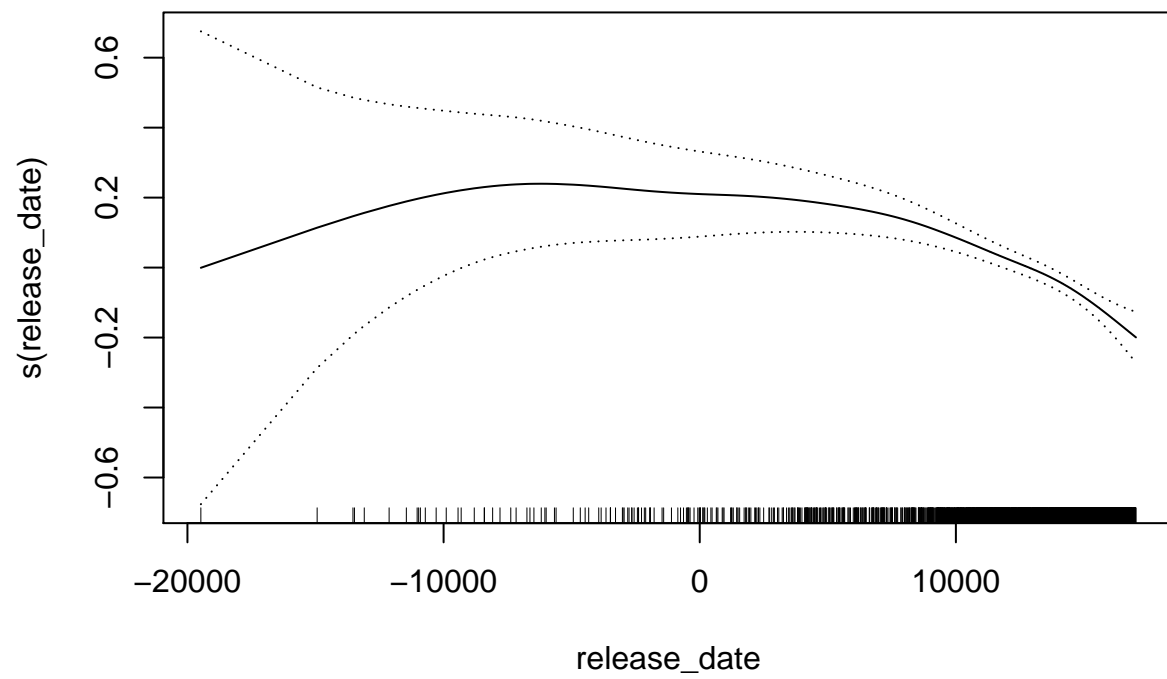
## Comp_Miramax          1  0.82  0.819  2.0009 0.1573446
## Comp_Pixar            1  5.66  5.661 13.8271 0.0002054 ***
## Comp_RelMedia         1  0.02  0.024  0.0576 0.8103679
## Comp_MGM              1  0.00  0.001  0.0022 0.9624321
## Comp_DreamWorks       1 12.57 12.574 30.7116 3.351e-08 ***
## Comp_Touchstone       1  0.06  0.062  0.1503 0.6982776
## Comp_CanalPlus        1  0.42  0.421  1.0290 0.3105035
## genere                18 109.14  6.063 14.8092 < 2.2e-16 ***
## Residuals             2204 902.38  0.409
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##              Npar Df Npar F      Pr(F)
## (Intercept)
## s(budget)          3 32.801 < 2.2e-16 ***
## s(popularity)      3 69.598 < 2.2e-16 ***
## s(release_date)    3  5.607 0.0007922 ***
## s(revenue)         3  6.176 0.0003544 ***
## s(runtime)         3 21.057 1.892e-13 ***
## production_countries
## spoken_languages
## Comp_Universal
## Comp_Paramount
## Comp_Warner
## Comp_20fox
## Comp_Columbia
## Comp_NewLine
## Comp_Disney
## Comp_VillageRoadshow
## Comp_Miramax
## Comp_Pixar
## Comp_RelMedia
## Comp_MGM
## Comp_DreamWorks
## Comp_Touchstone
## Comp_CanalPlus
## genere
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

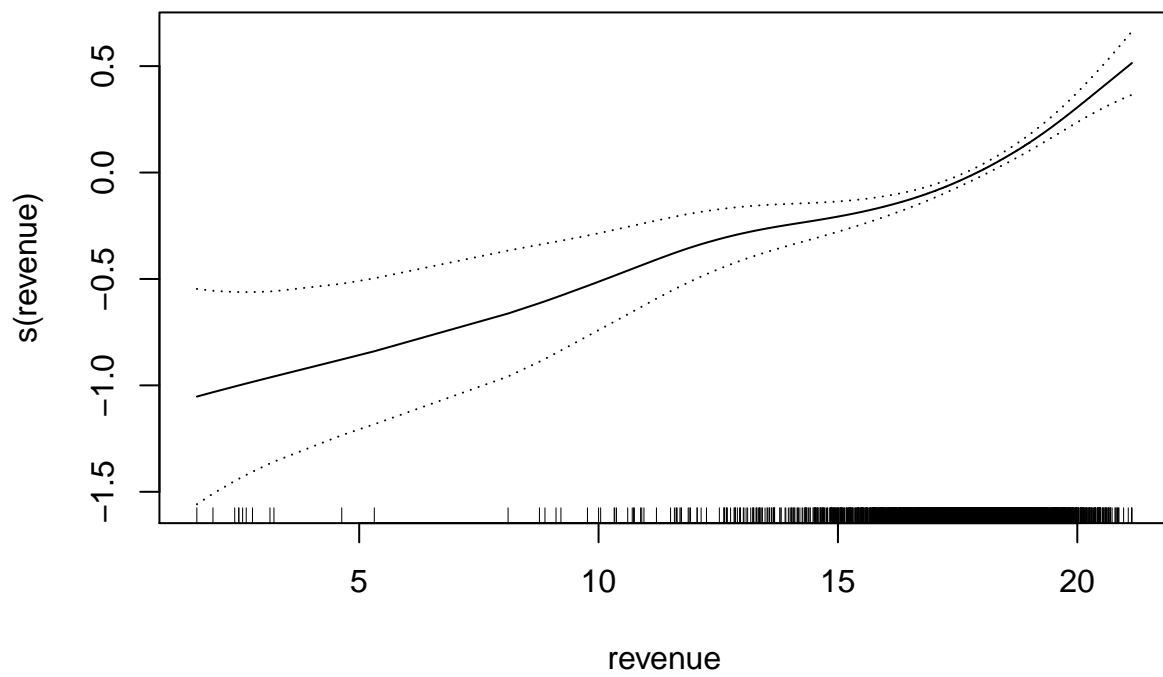
```

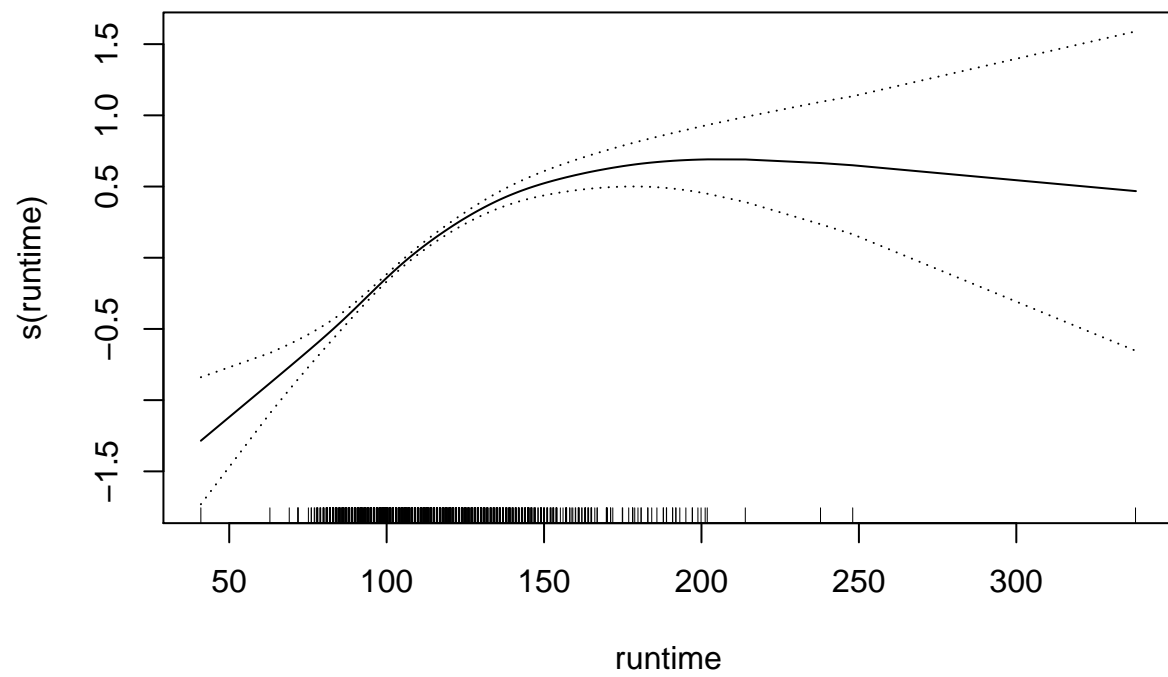
```
plot(g1, se=T, ask=F)
```

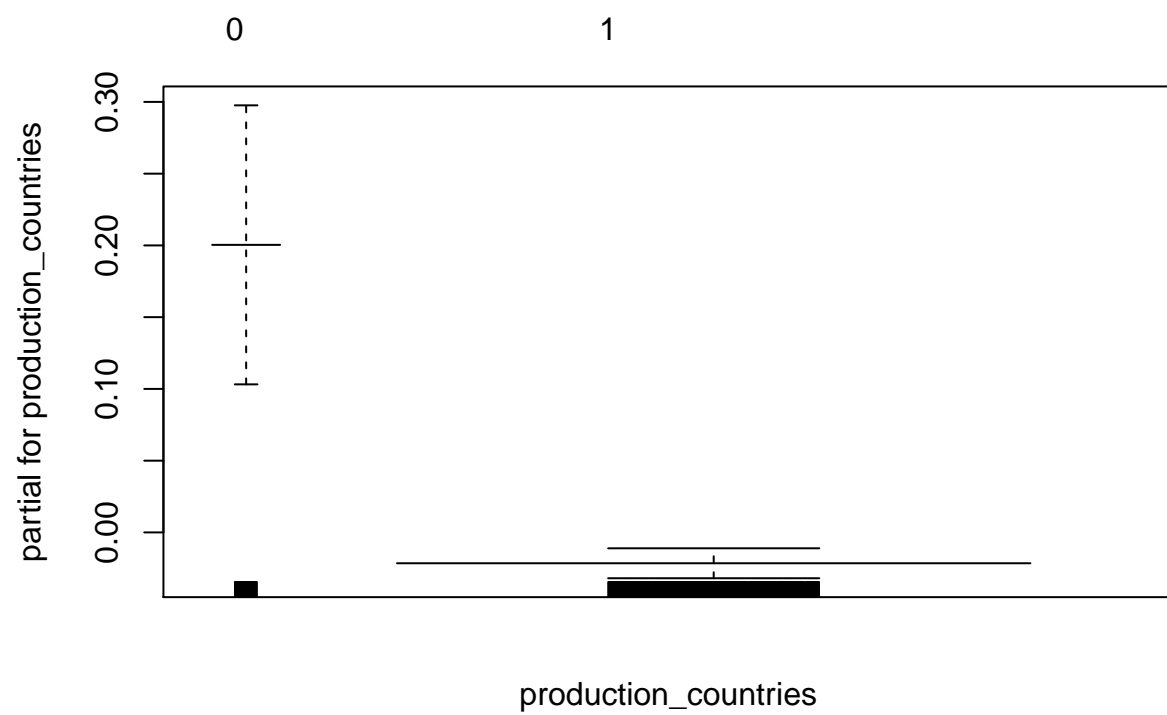


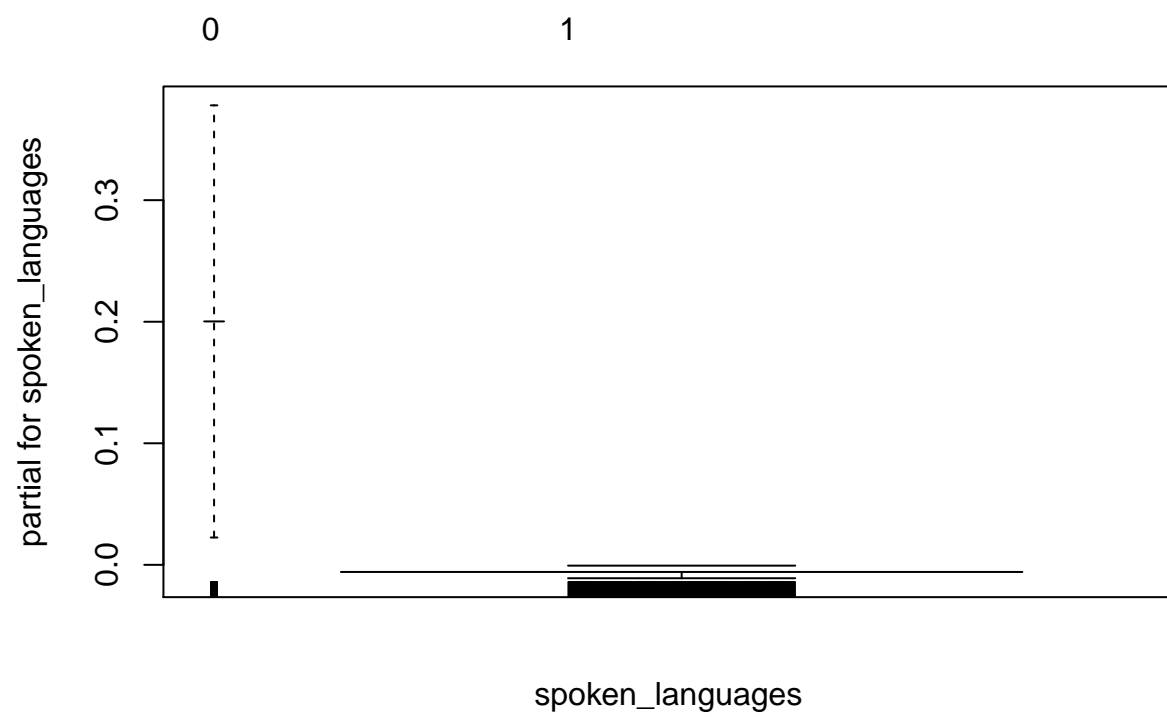


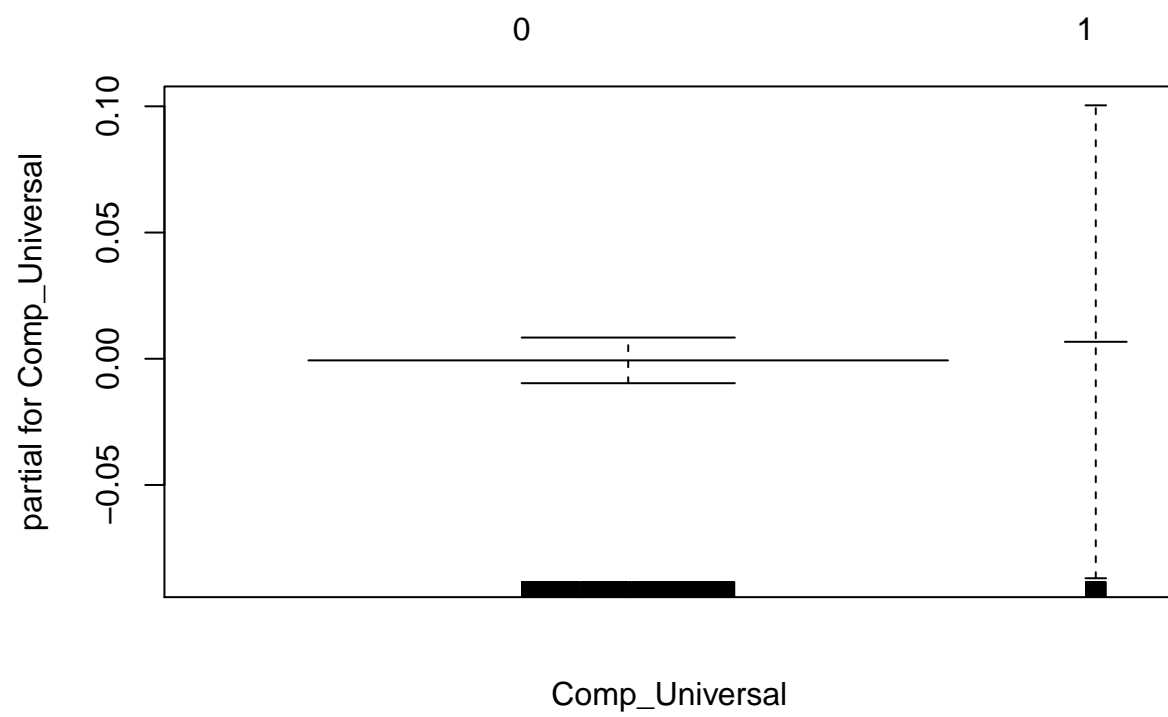


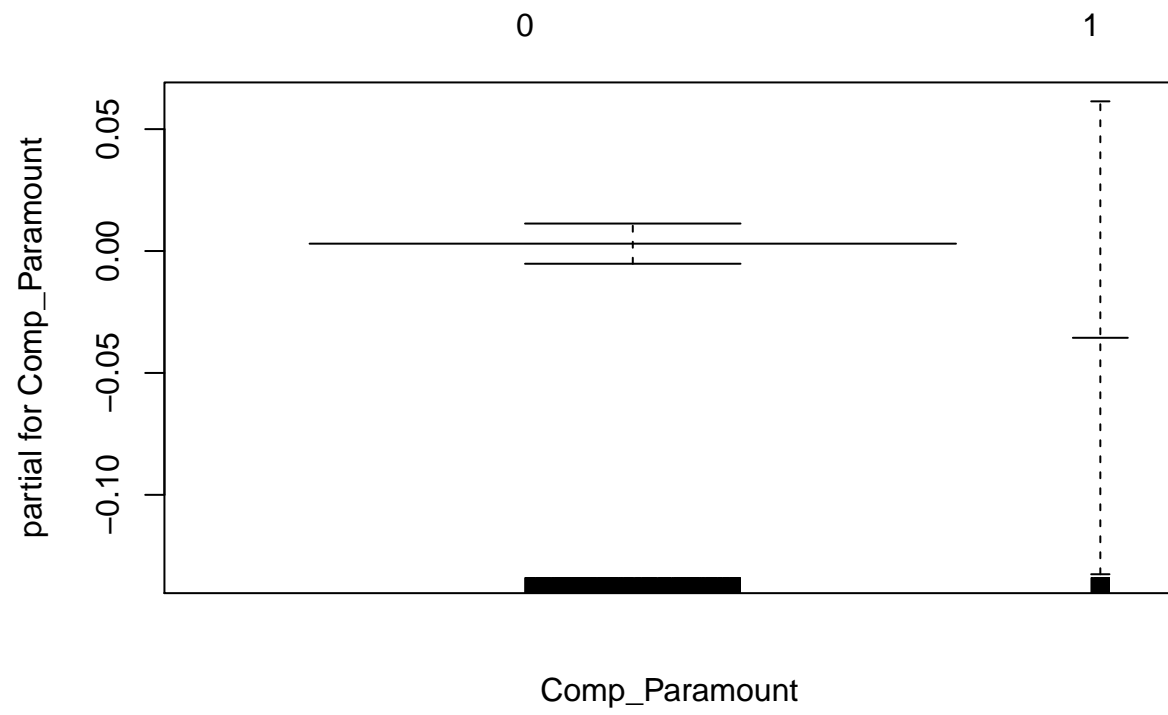


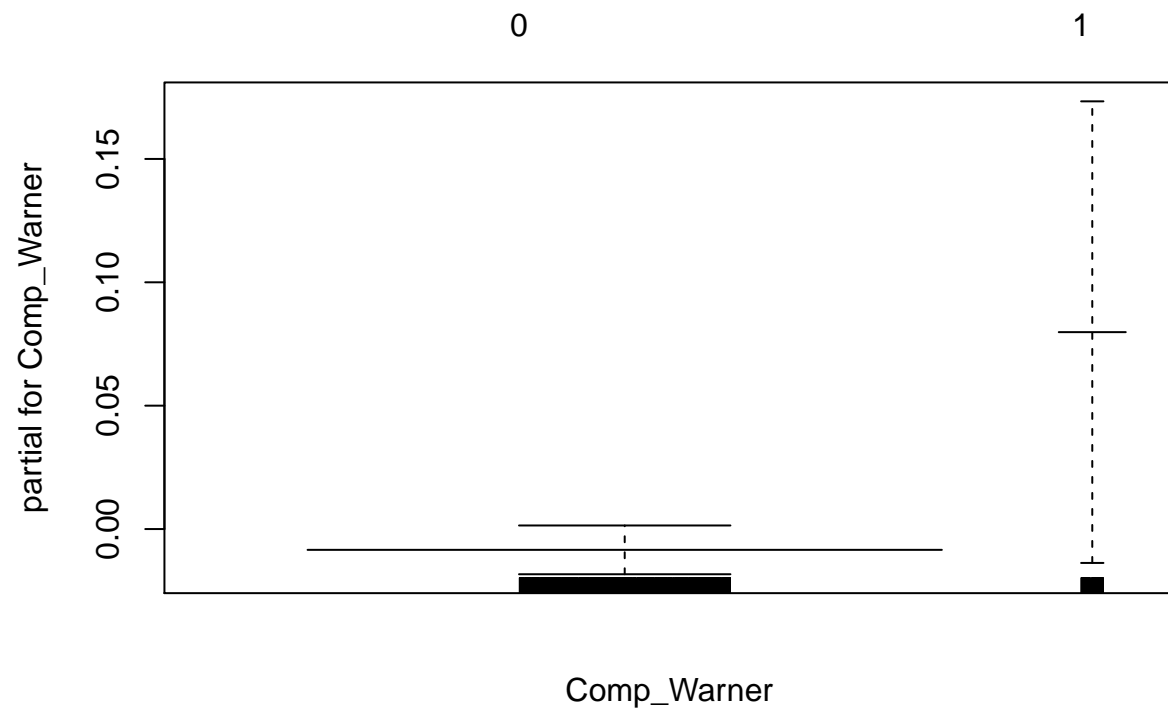


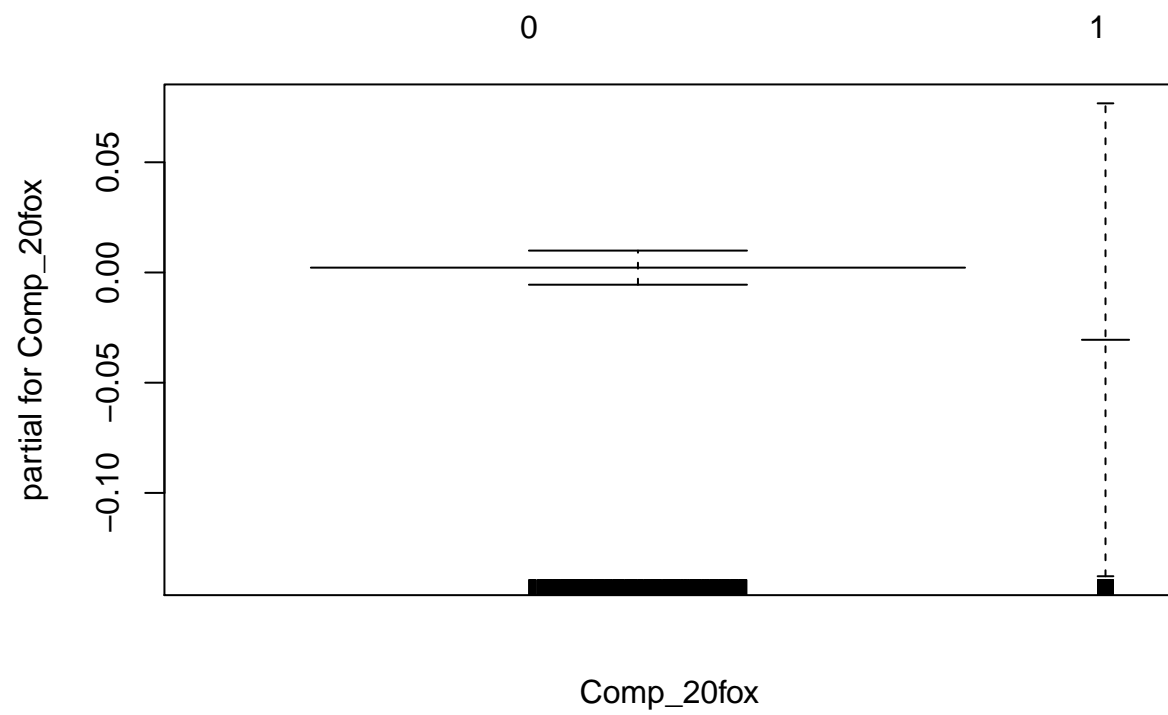


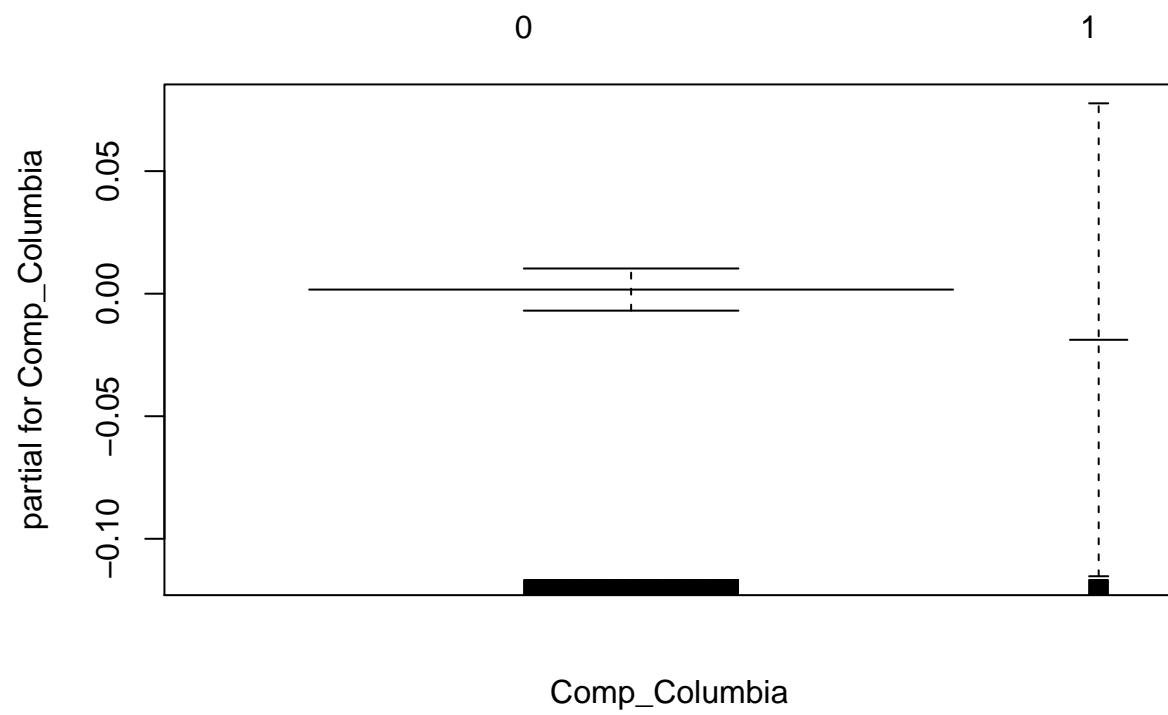


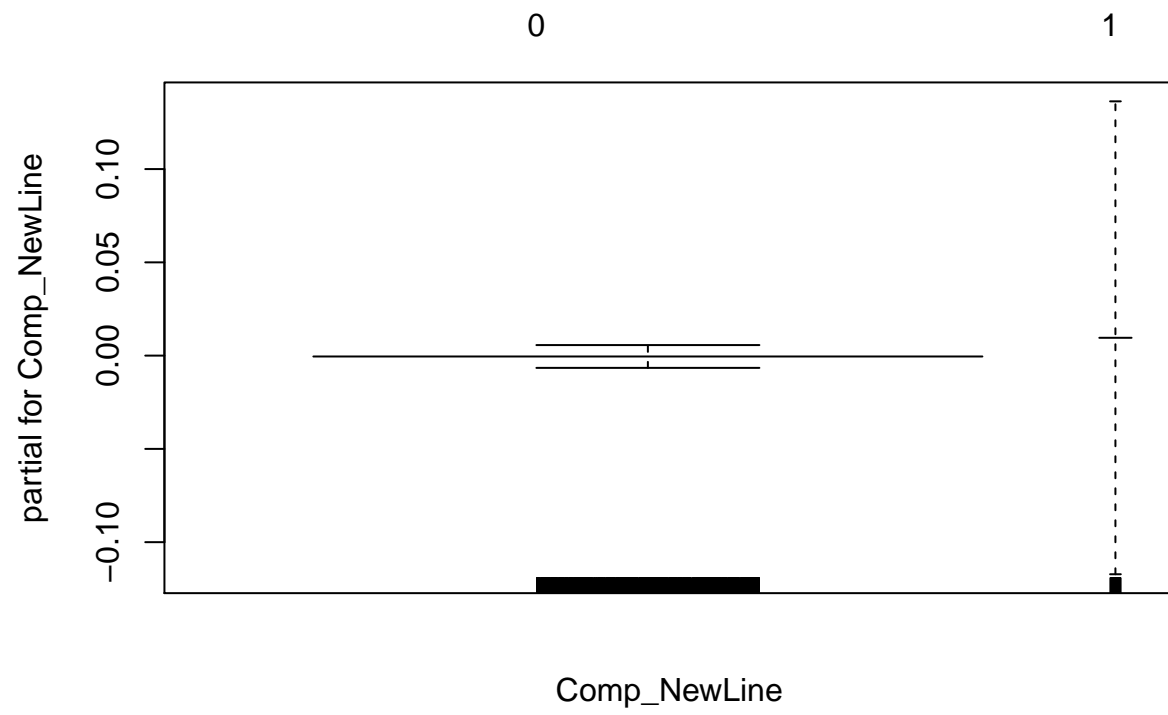


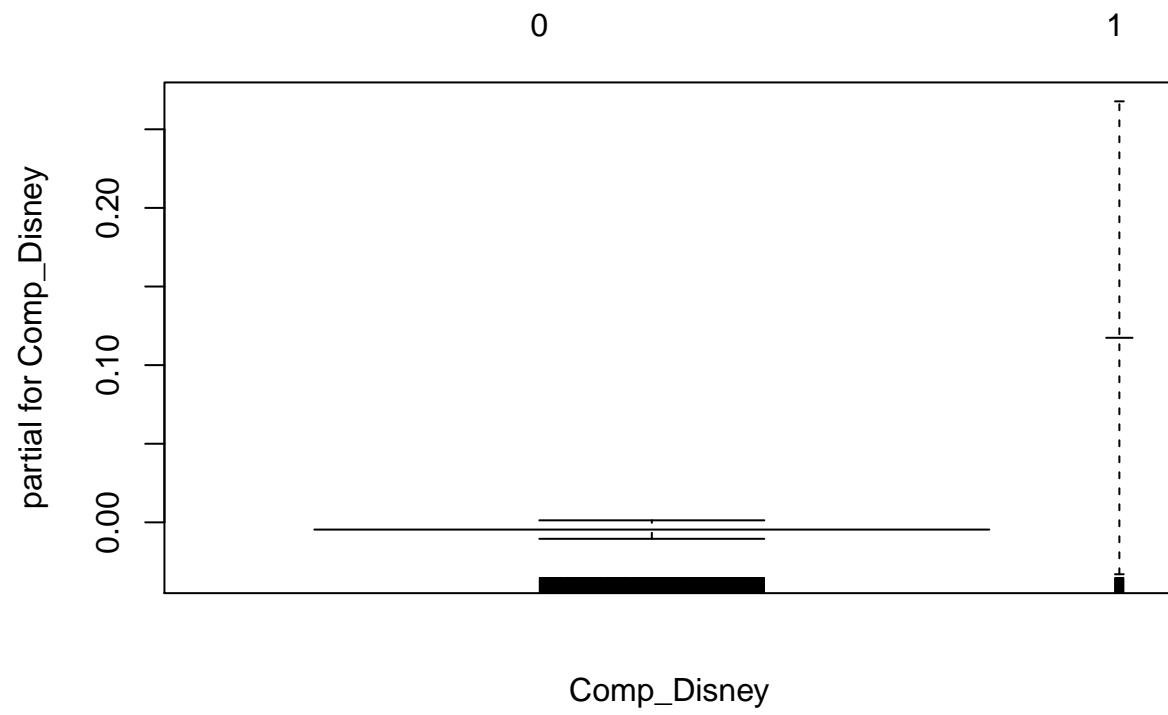


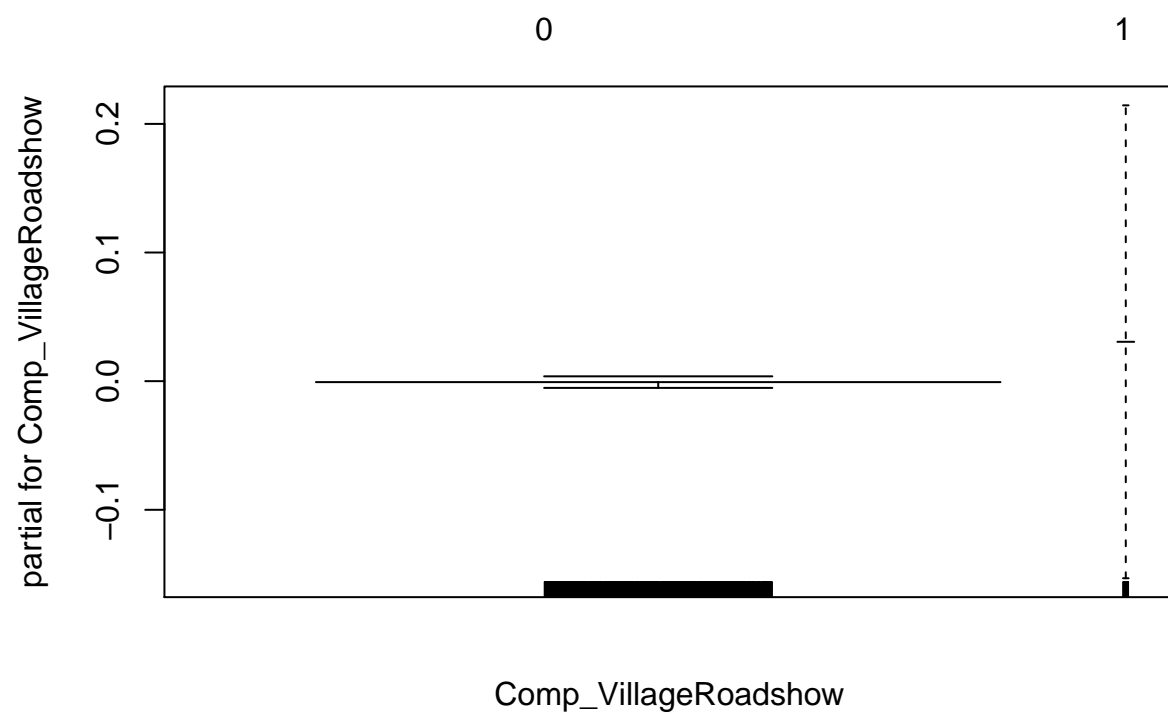


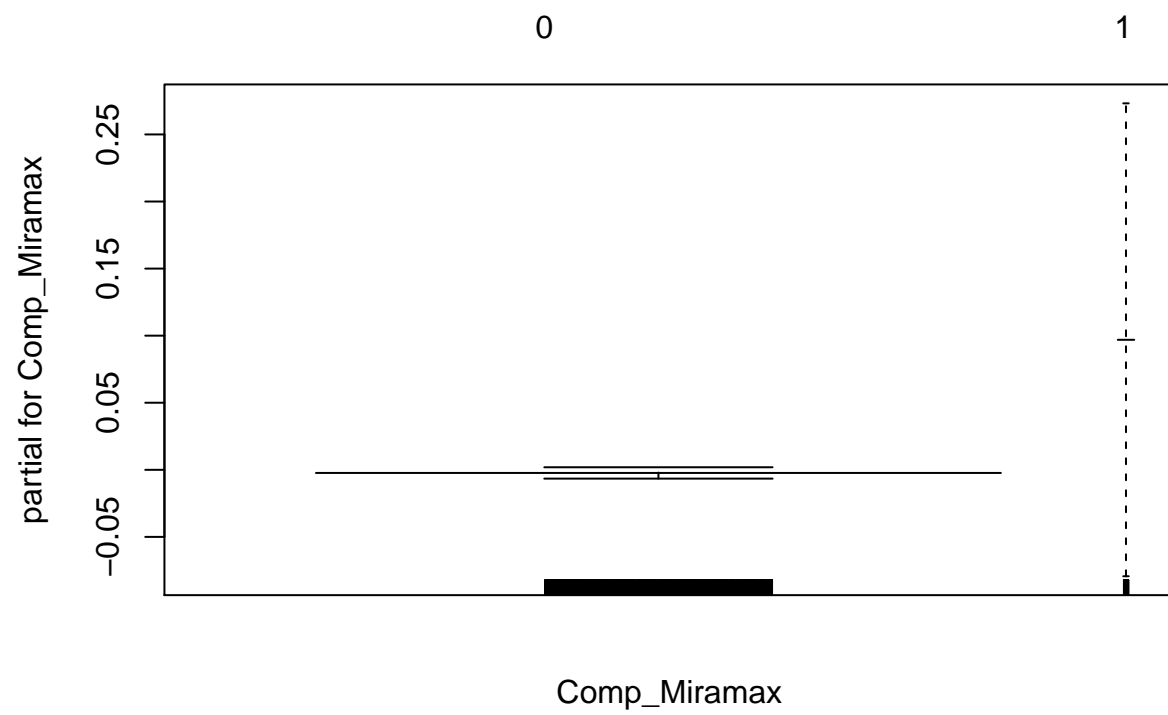


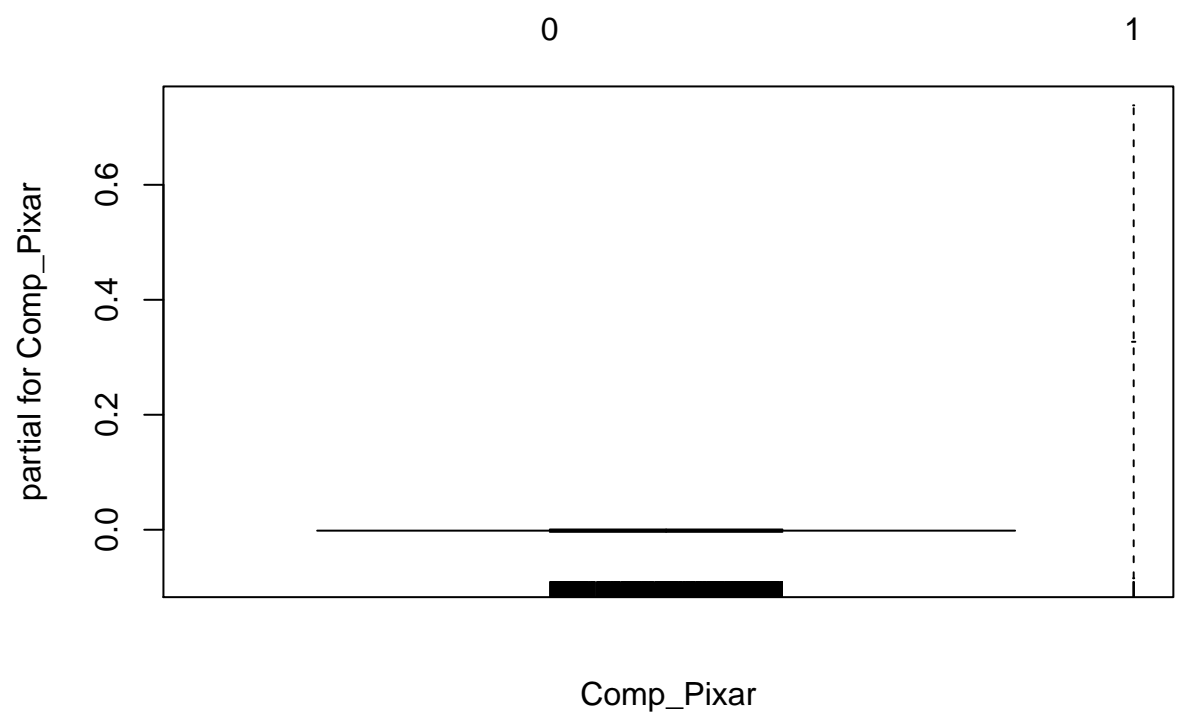


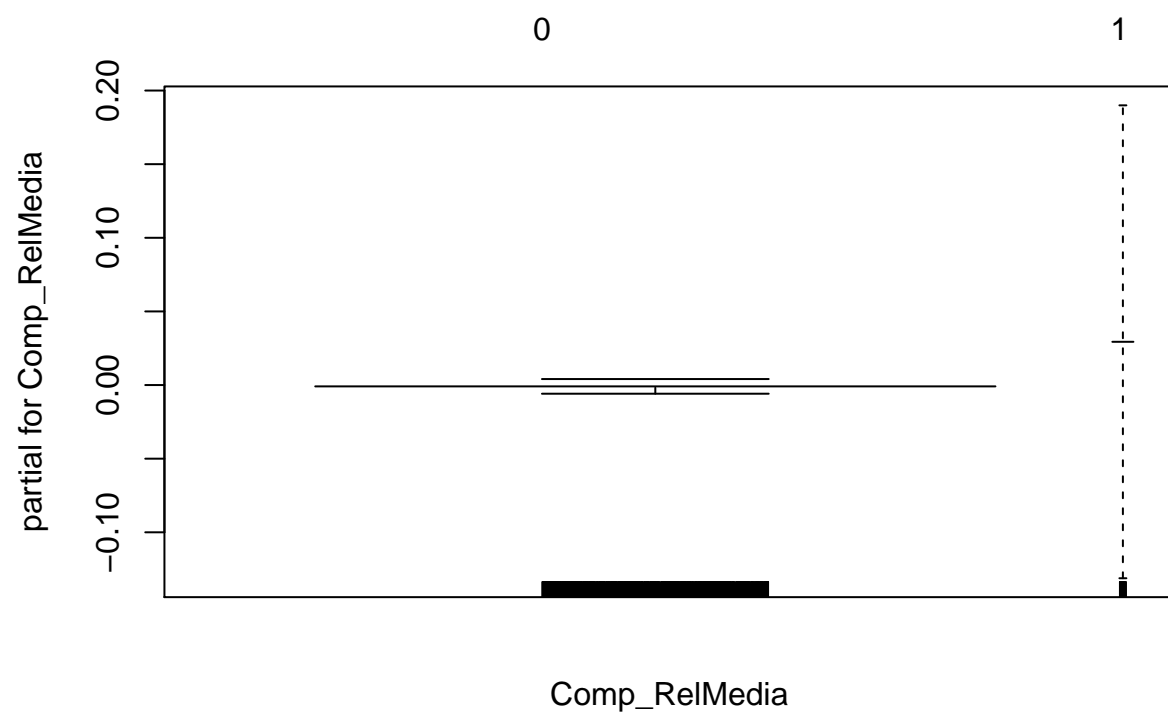


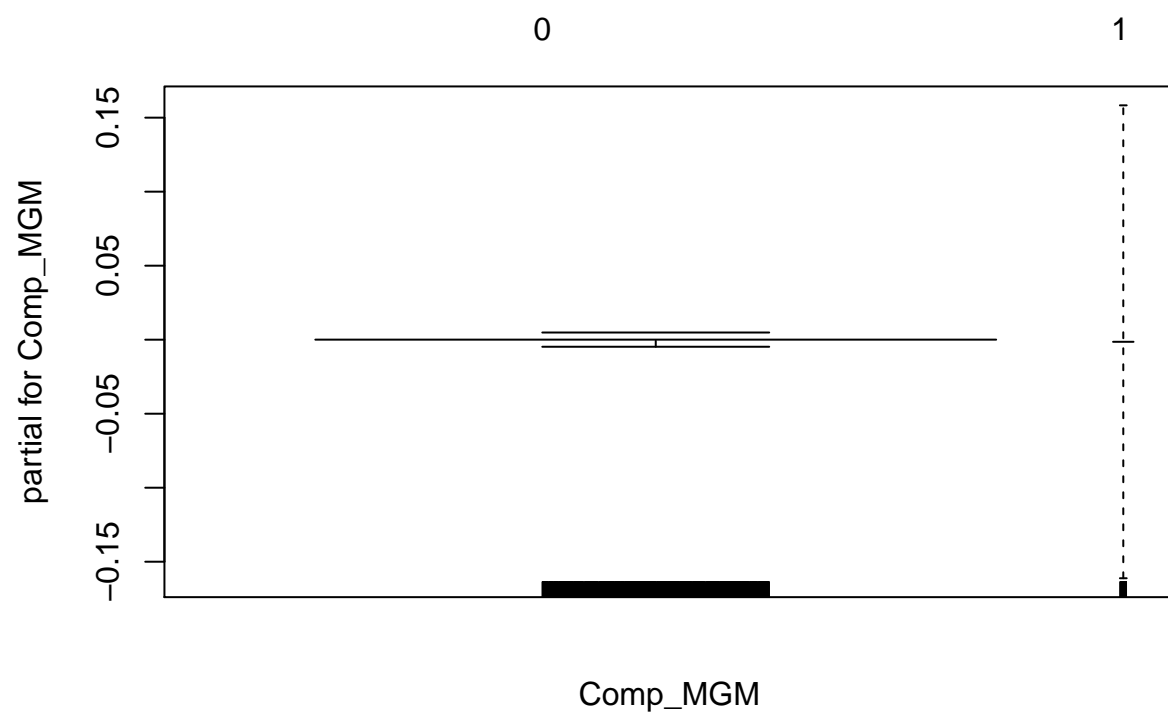


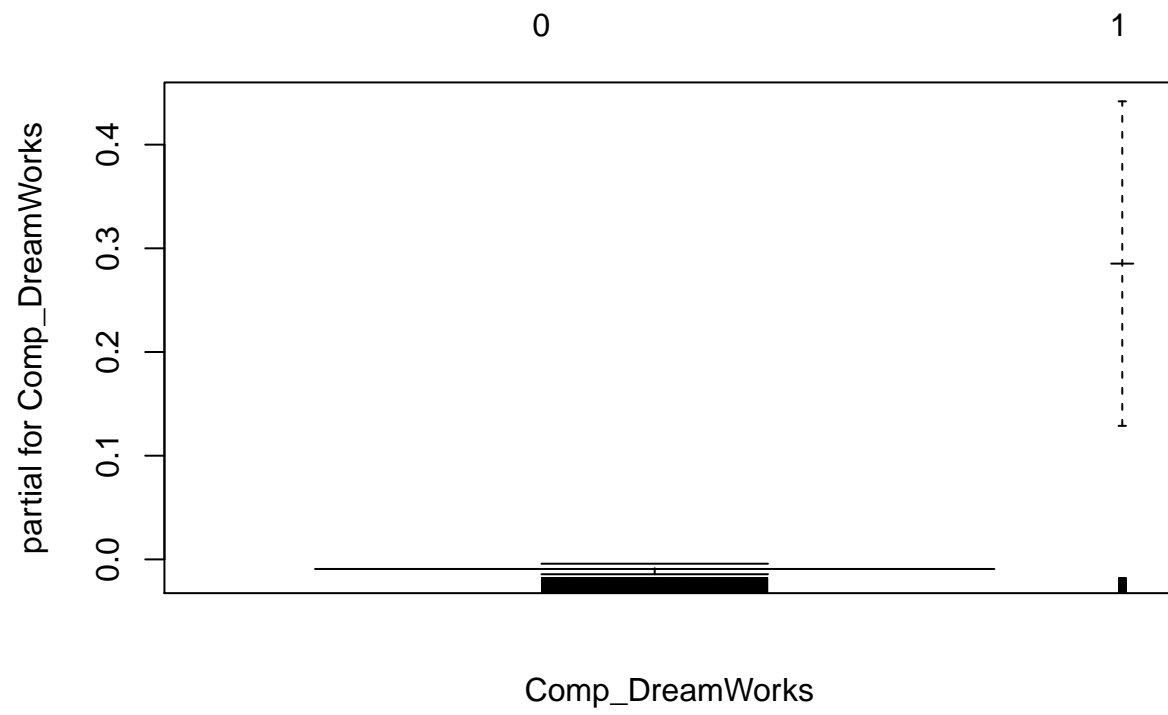


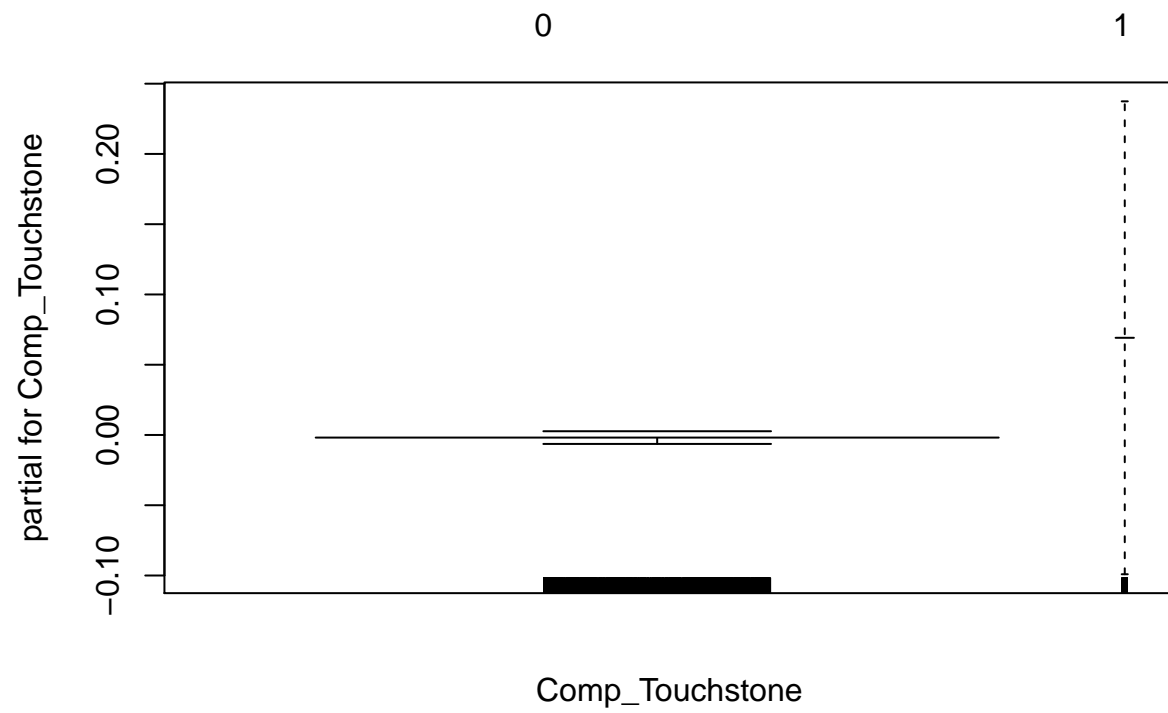


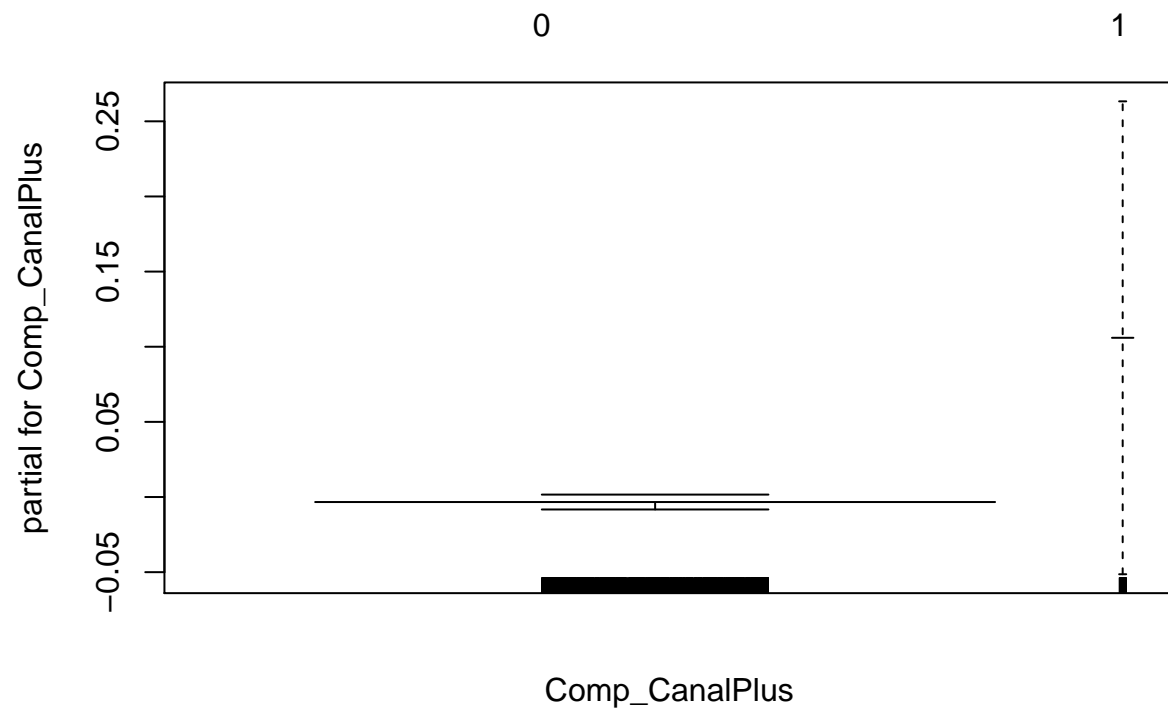


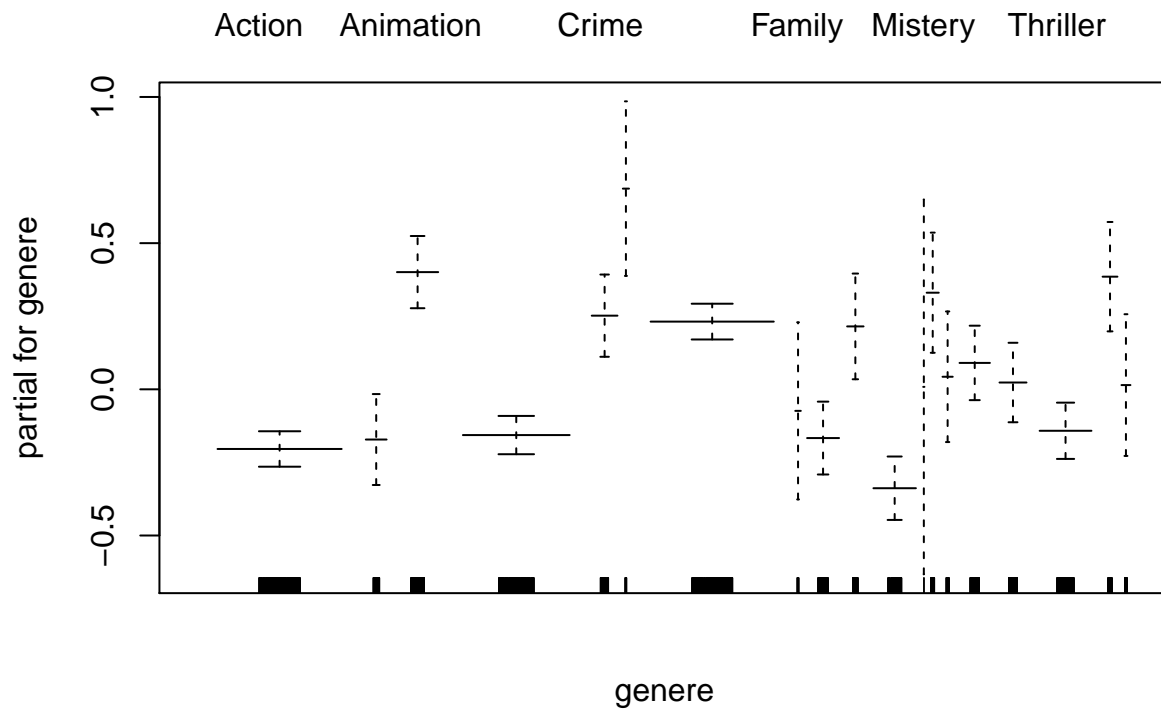












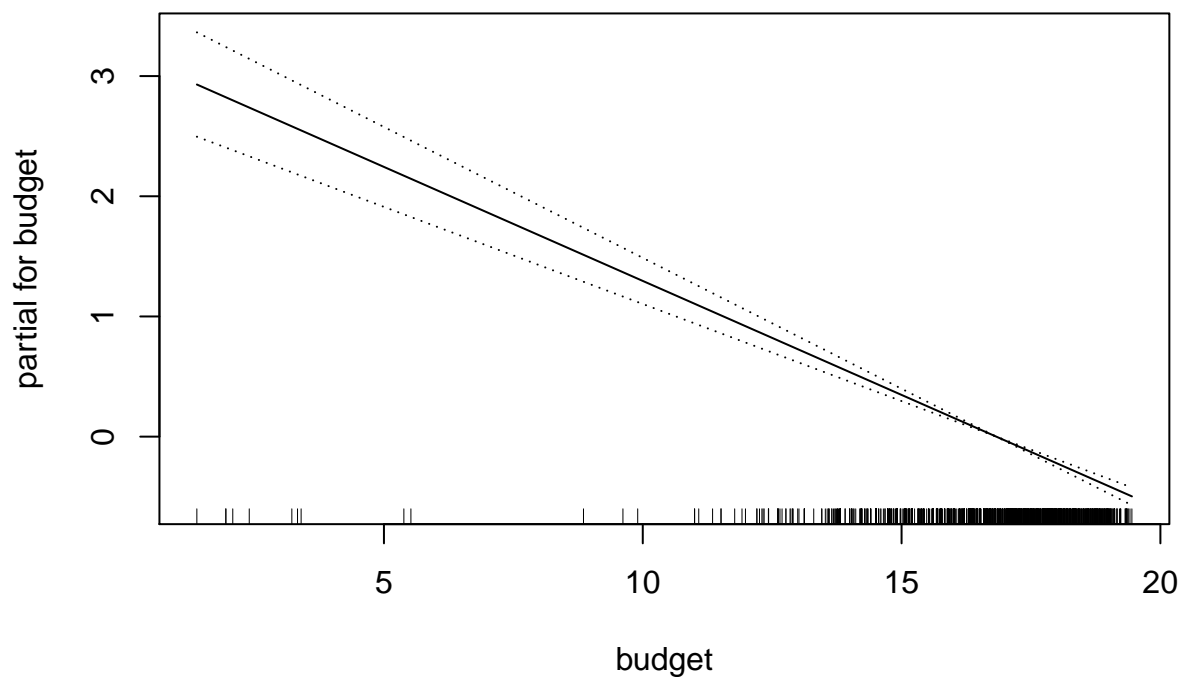
To perform a stepwise model selection with GAM, we have to start with a linear model (df=1). This below is a linear model (no splines, no loess and so on)

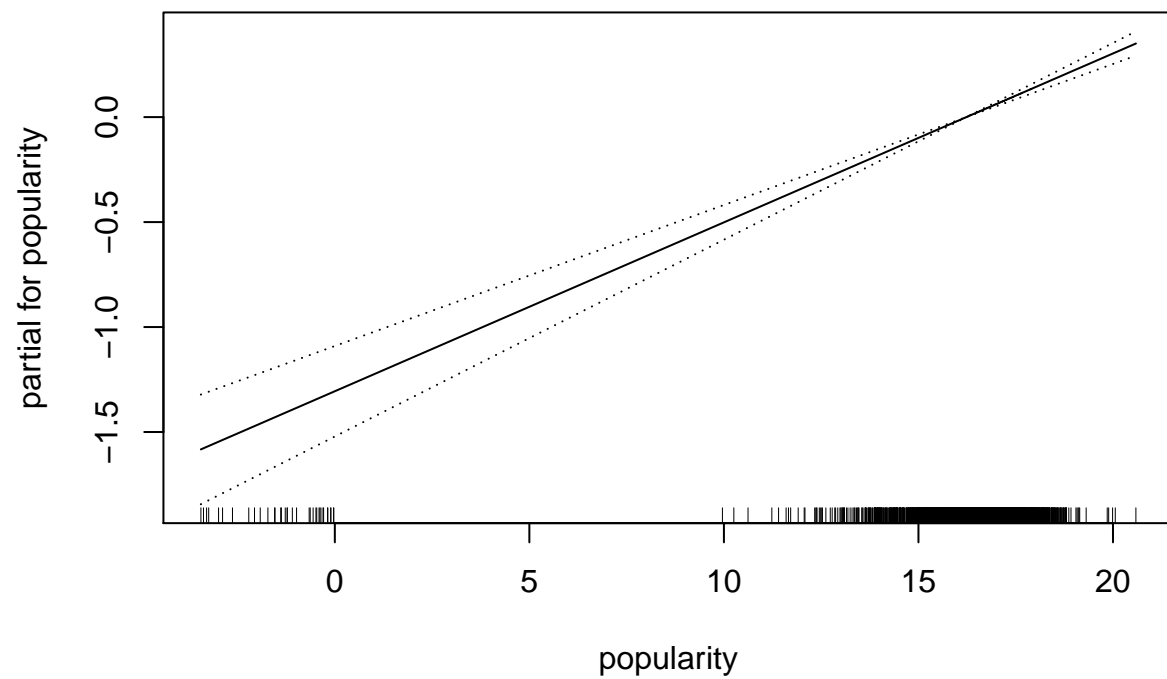
```
g2 <- gam(vote_average~.-vote_classes, data=dati.train)
#We have only the parametric part
summary(g2)
```

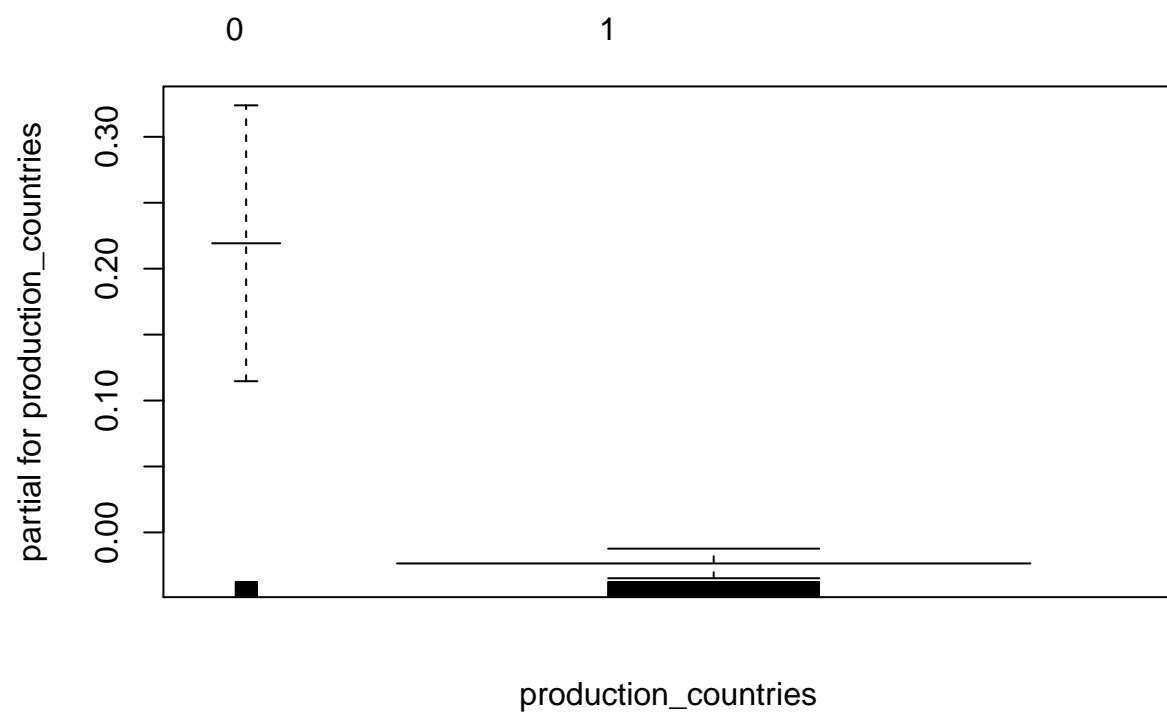
```
##
## Call: gam(formula = vote_average ~ . - vote_classes, data = dati.train)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -4.40486 -0.39621  0.03864  0.43335  4.23002
##
## (Dispersion Parameter for gaussian family taken to be 0.4739)
##
##      Null Deviance: 1696.602 on 2259 degrees of freedom
## Residual Deviance: 1051.682 on 2219 degrees of freedom
## AIC: 4768.761
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##              Df  Sum Sq Mean Sq  F value    Pr(>F)
## budget          1    25.00   25.005   52.7589 5.195e-13 ***
## popularity       1   122.00  122.000  257.4137 < 2.2e-16 ***
## production_countries 1    20.28   20.278   42.7847 7.562e-11 ***
```

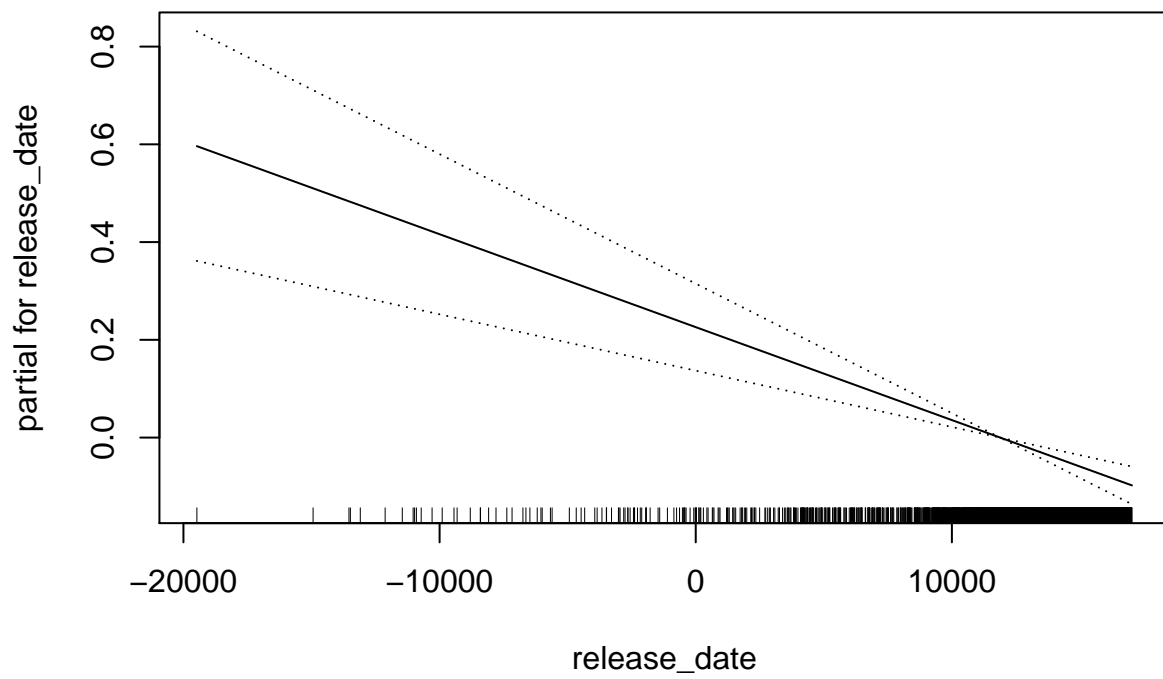
```
## release_date      1    62.31   62.306 131.4635 < 2.2e-16 ***
## revenue           1    78.26   78.263 165.1317 < 2.2e-16 ***
## runtime           1   194.65  194.650 410.7028 < 2.2e-16 ***
## spoken_languages   1     2.24    2.243   4.7322  0.02971 *
## Comp_Universal     1     0.66    0.663   1.3996  0.23692
## Comp_Paramount     1     2.05    2.050   4.3251  0.03767 *
## Comp_Warner        1     0.03    0.035   0.0733  0.78664
## Comp_20fox         1     1.83    1.828   3.8568  0.04967 *
## Comp_Columbia      1     1.01    1.009   2.1297  0.14461
## Comp_NewLine       1     1.99    1.987   4.1927  0.04072 *
## Comp_Disney        1     1.21    1.215   2.5626  0.10956
## Comp_VillageRoadshow 1     0.83    0.830   1.7516  0.18581
## Comp_Miramax       1     1.59    1.589   3.3534  0.06720 .
## Comp_Pixar         1     8.46    8.456  17.8420 2.497e-05 ***
## Comp_RelMedia      1     0.05    0.045   0.0954  0.75742
## Comp_MGM           1     0.18    0.180   0.3793  0.53805
## Comp_DreamWorks    1     7.51    7.513  15.8512 7.073e-05 ***
## Comp_Touchstone    1     0.04    0.044   0.0923  0.76128
## Comp_CanalPlus     1     0.07    0.068   0.1438  0.70457
## genere            18   112.66    6.259  13.2063 < 2.2e-16 ***
## Residuals         2219 1051.68    0.474
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

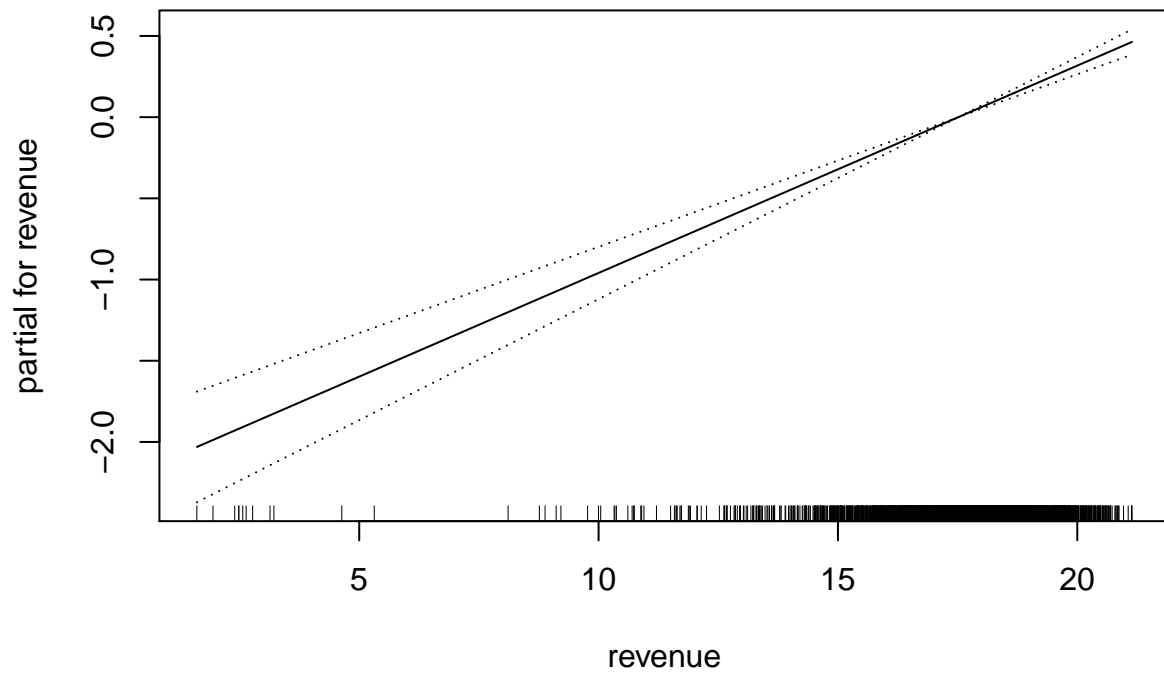
```
#Show the linear effects (in fact we have only straight lines)
plot(g2, se=T, ask=F)
```

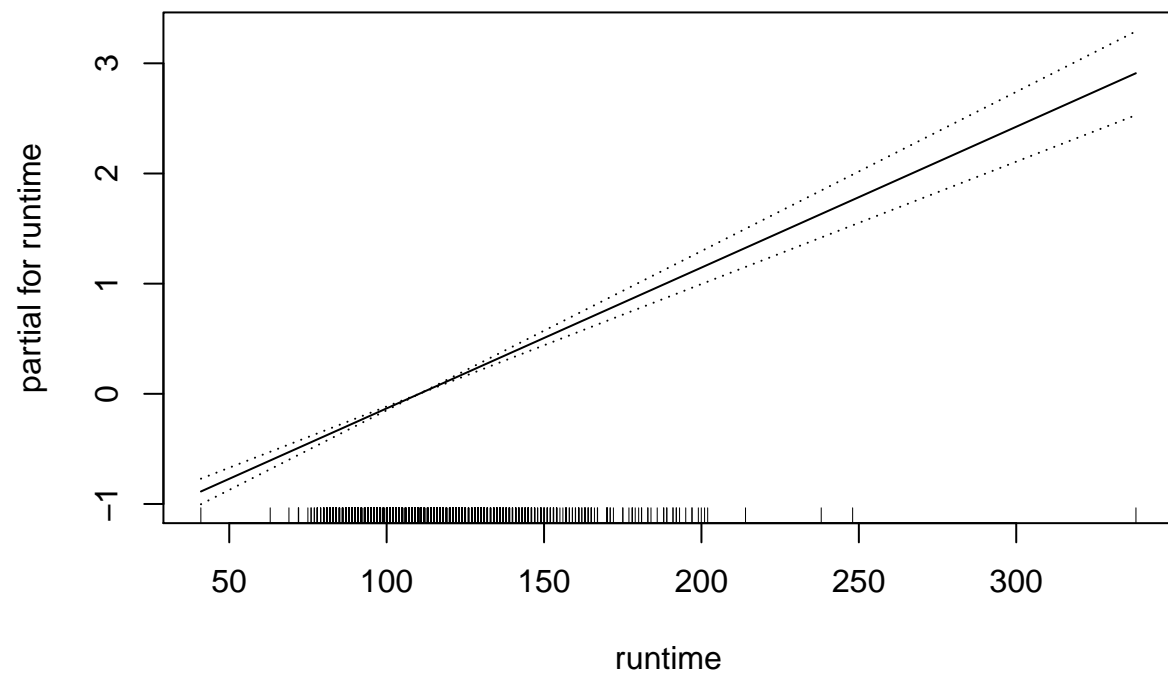


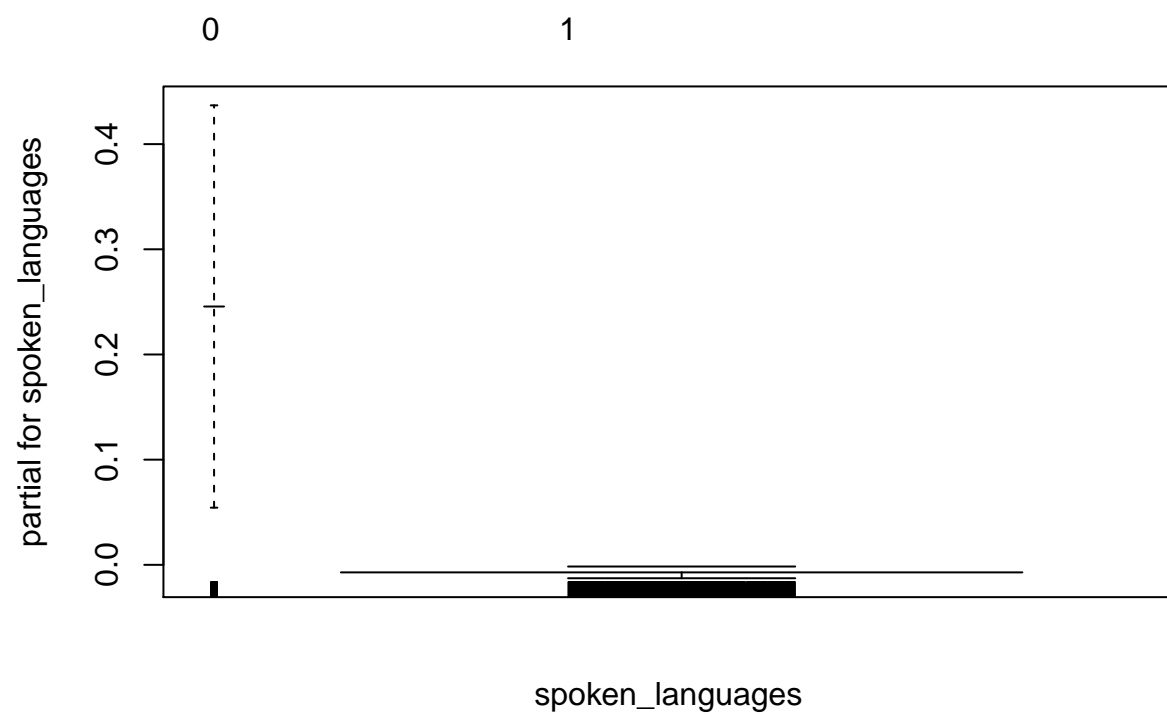


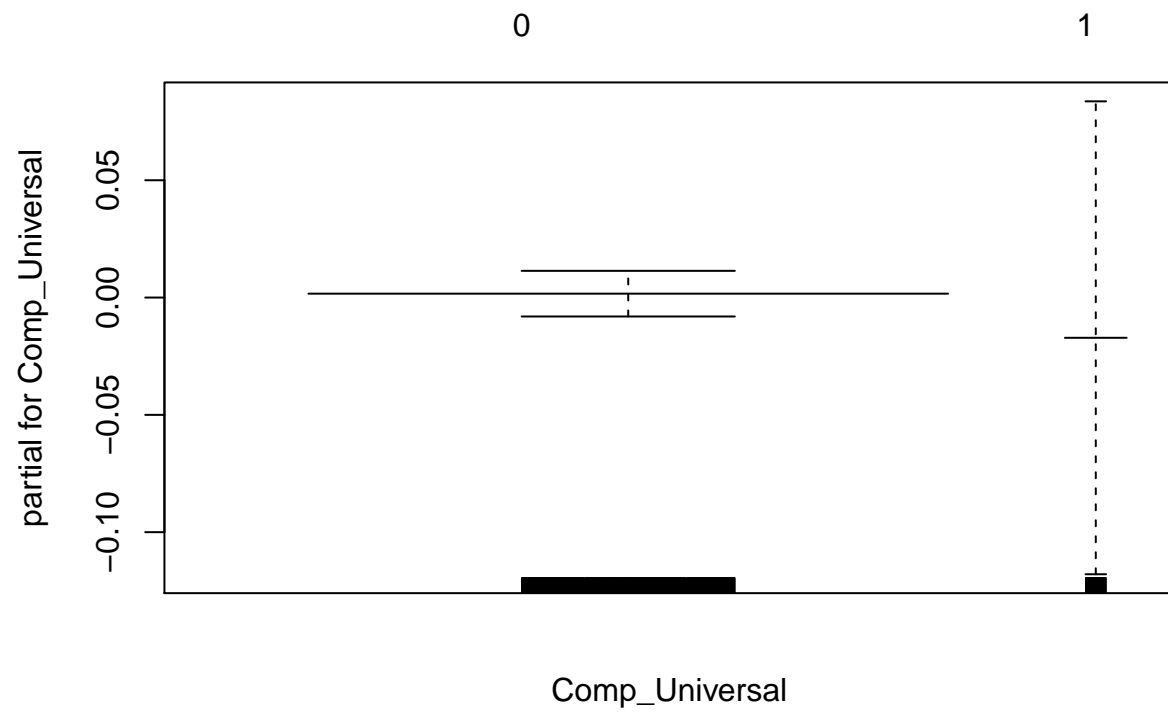


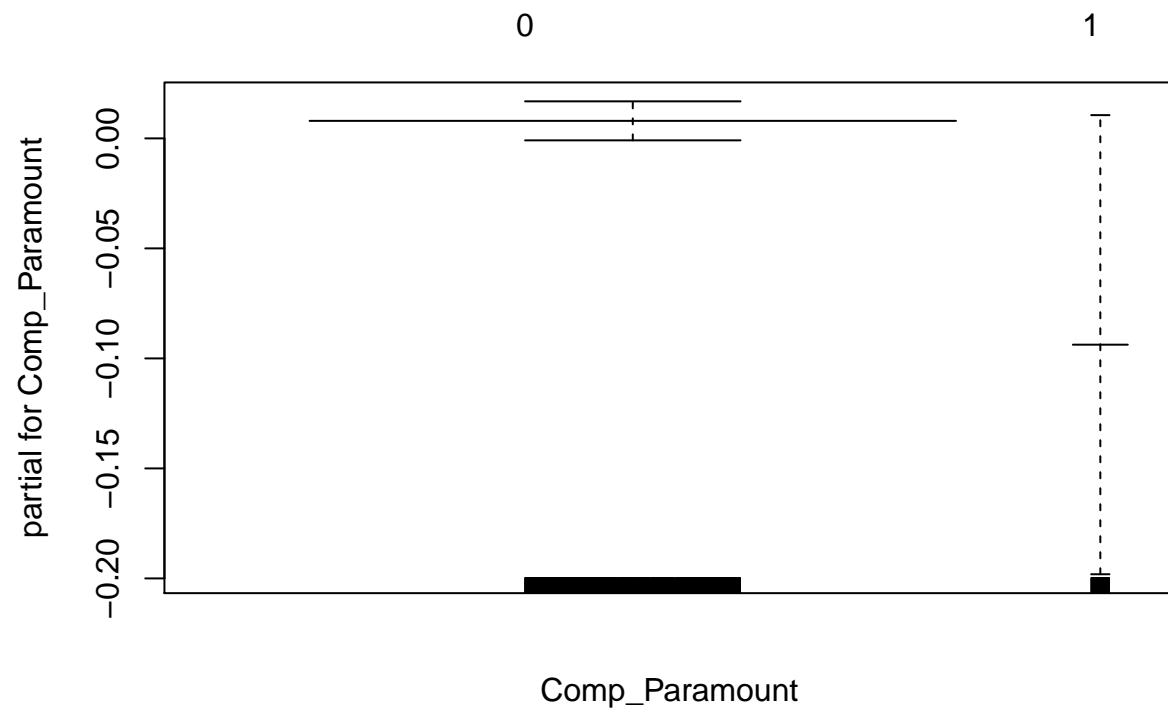


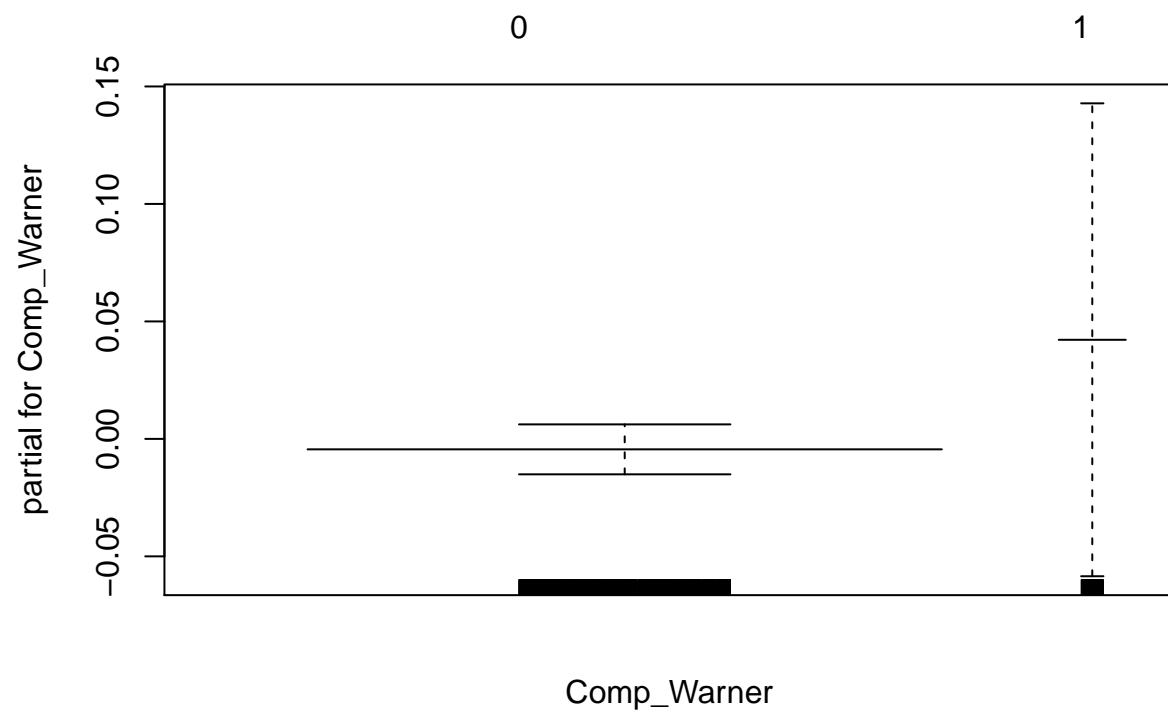


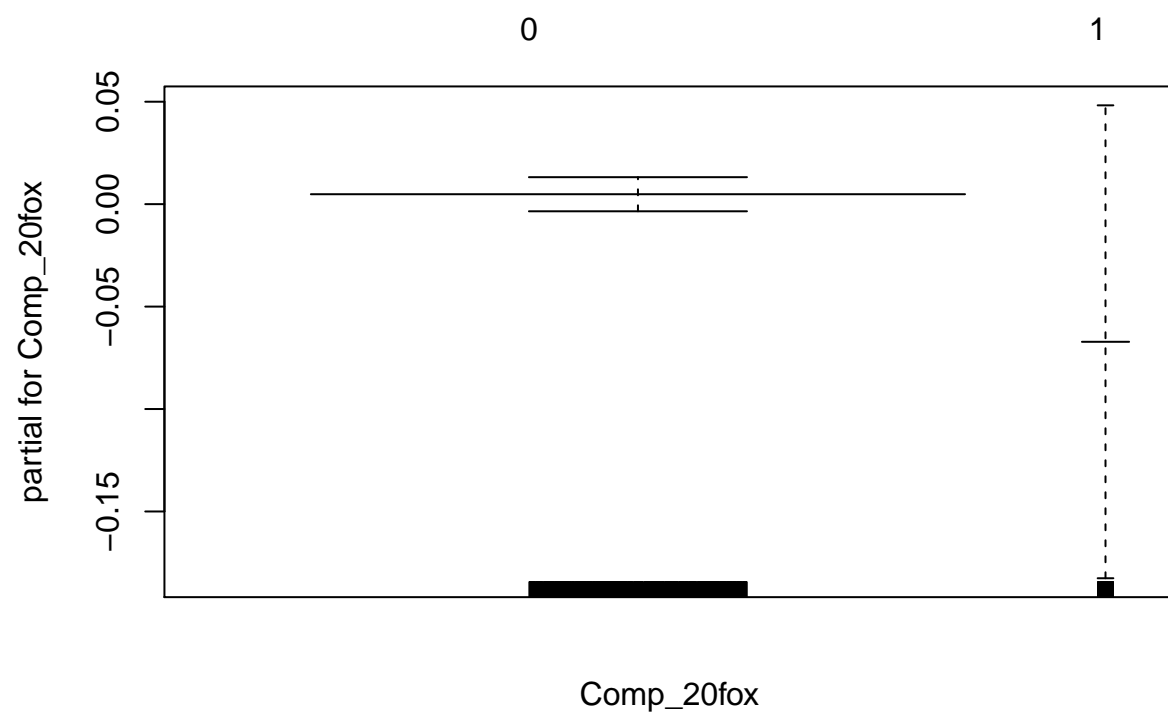


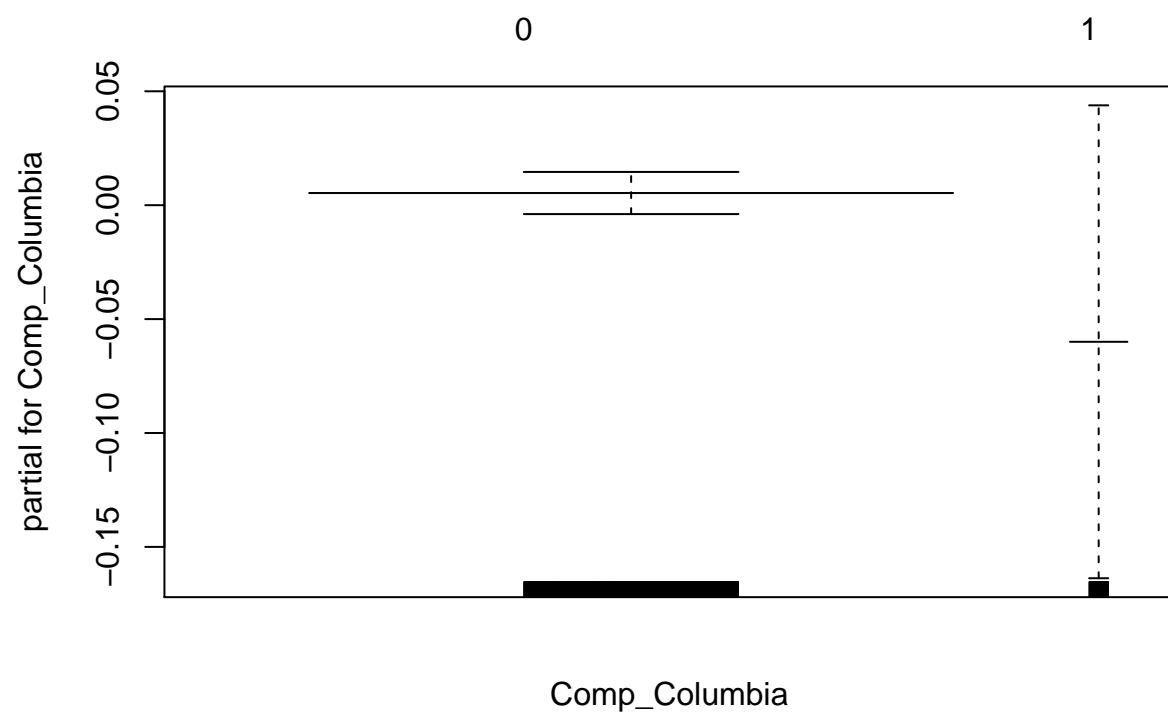


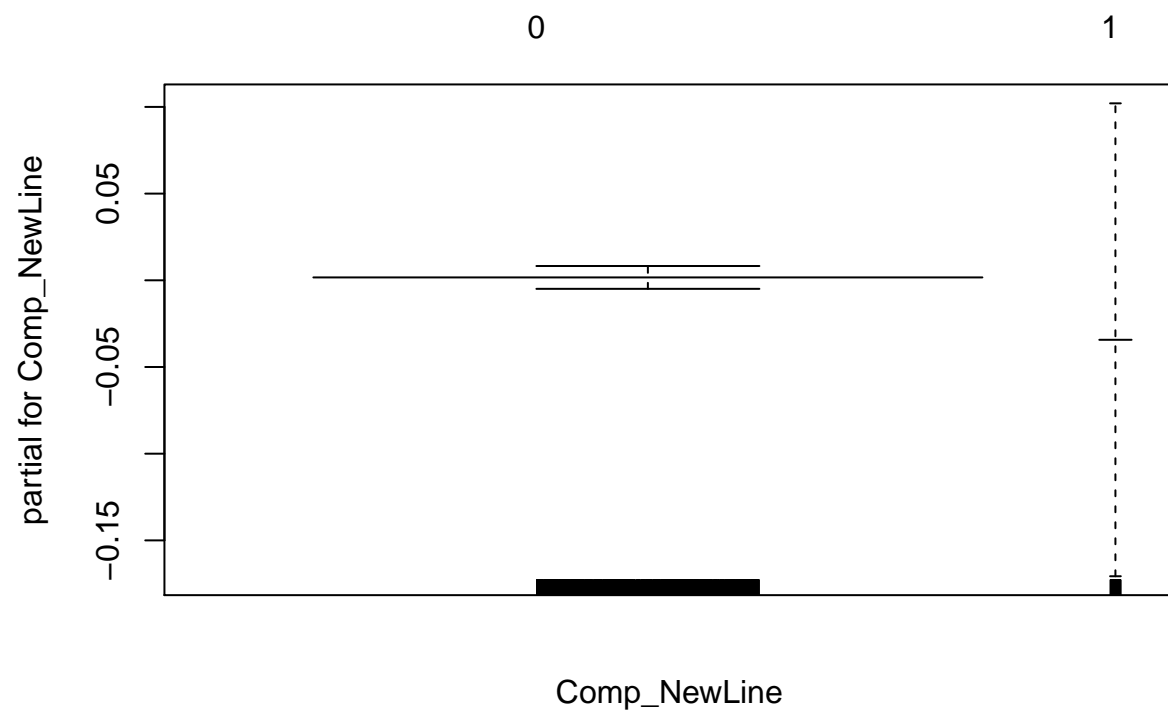


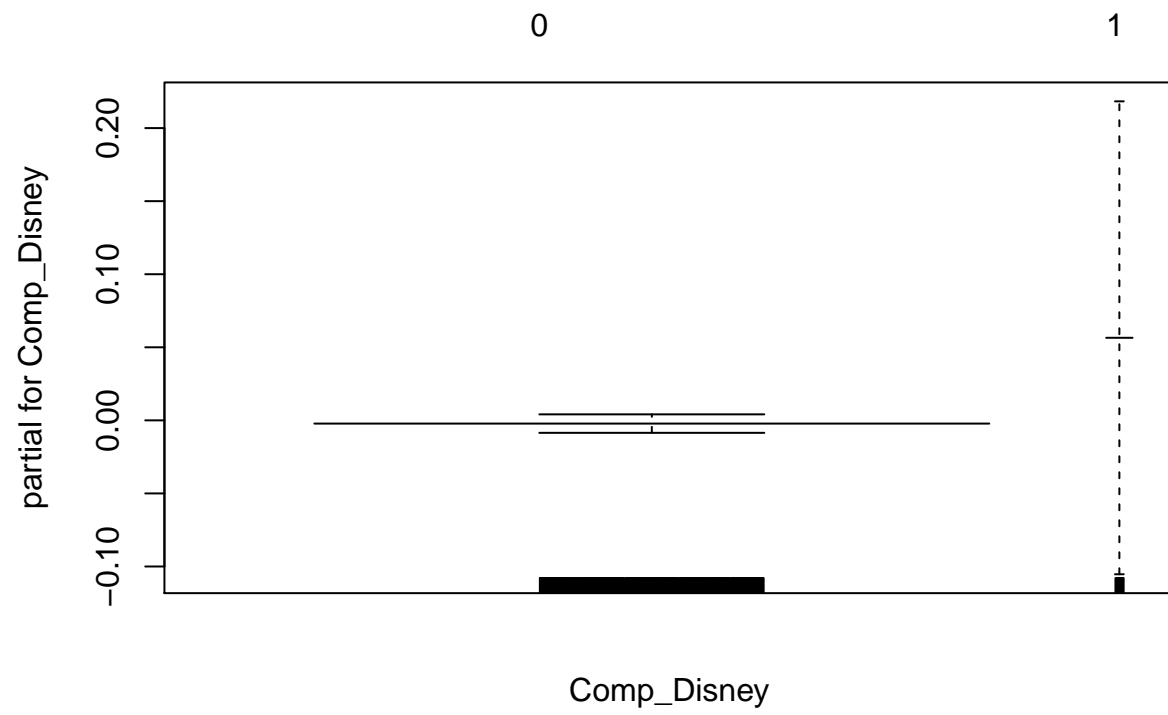


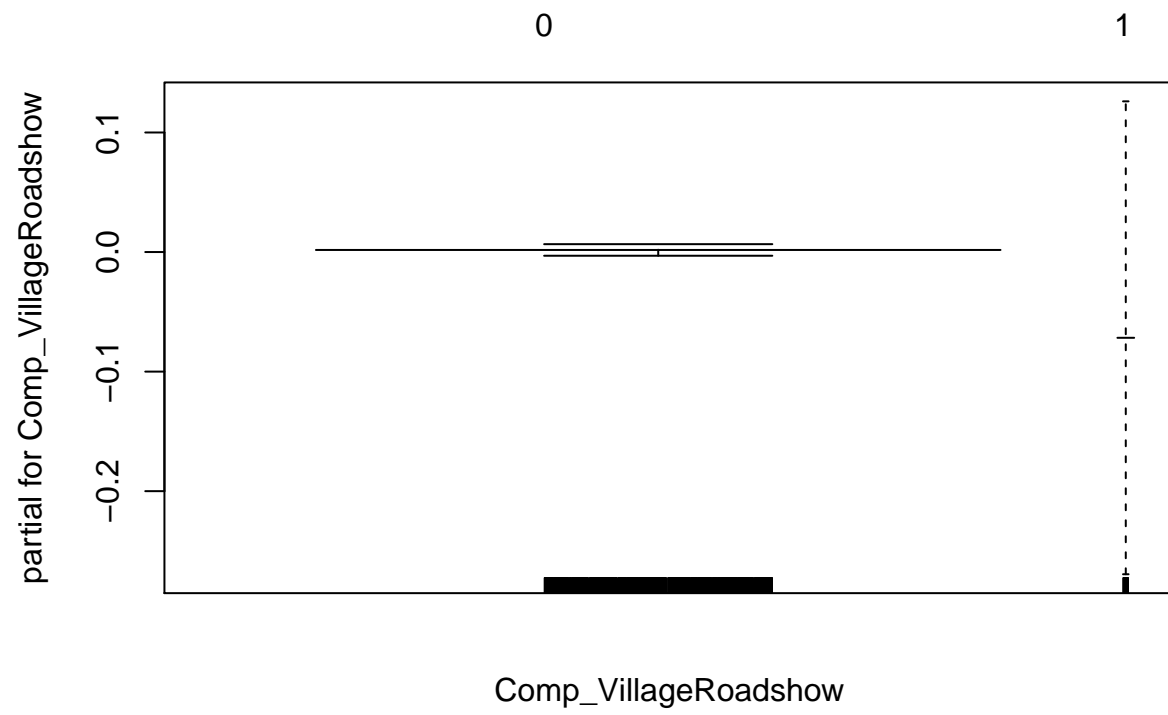


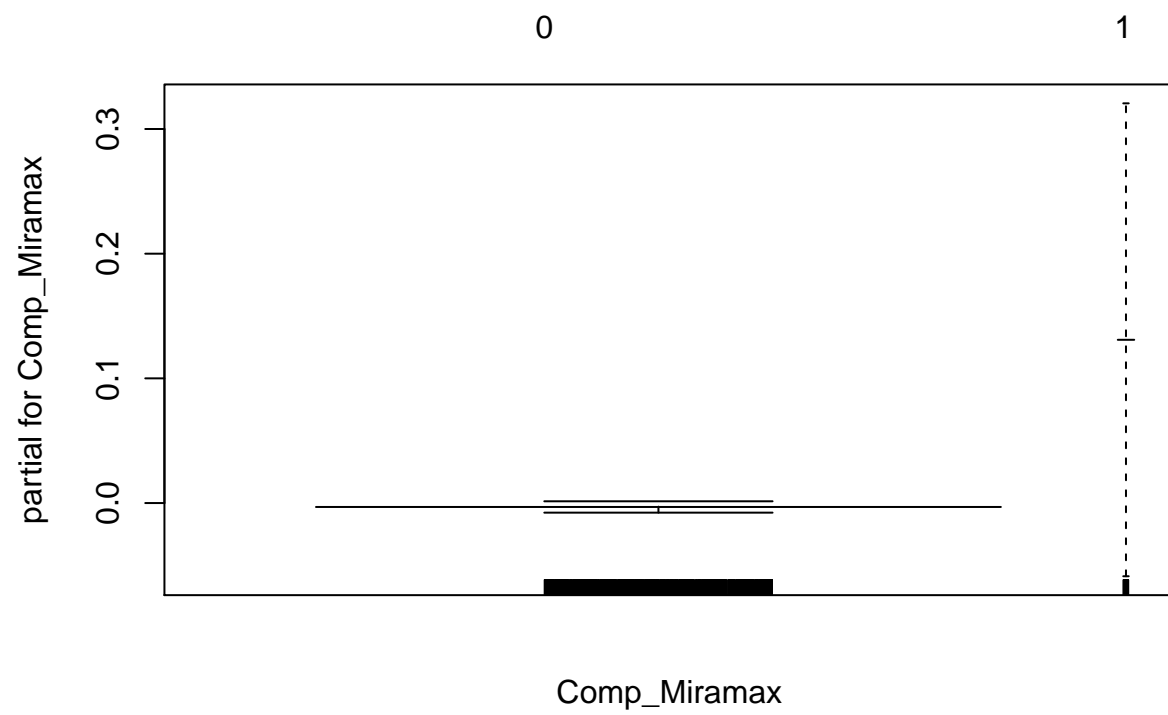


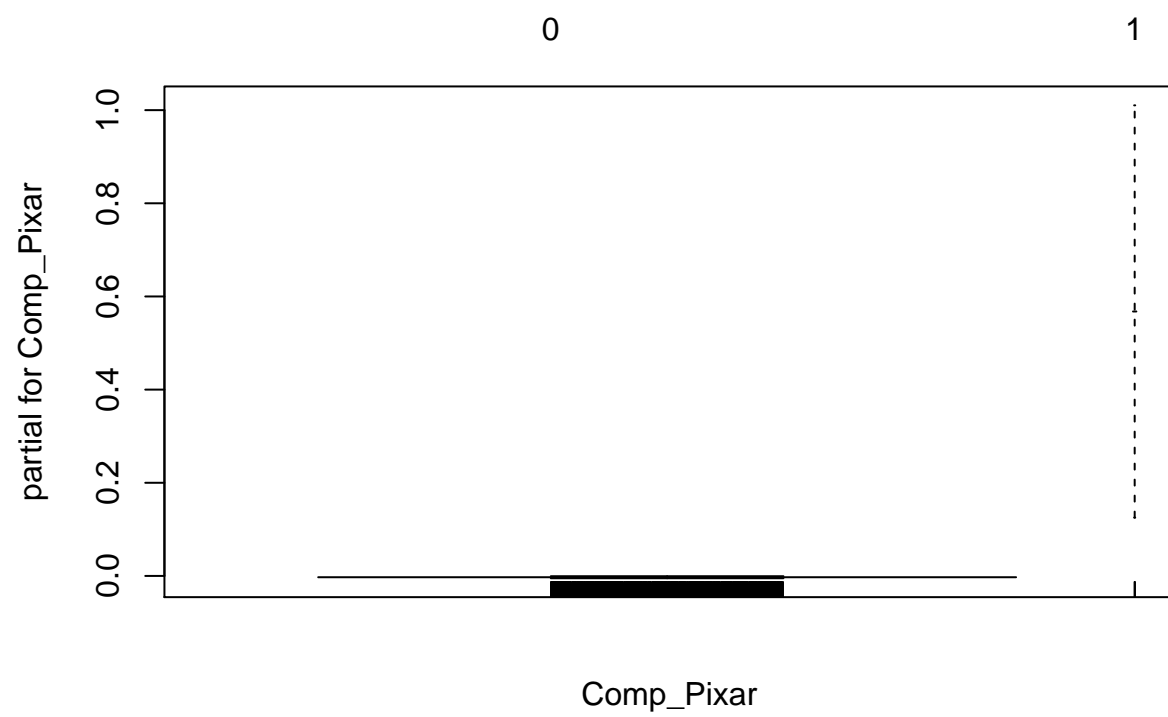


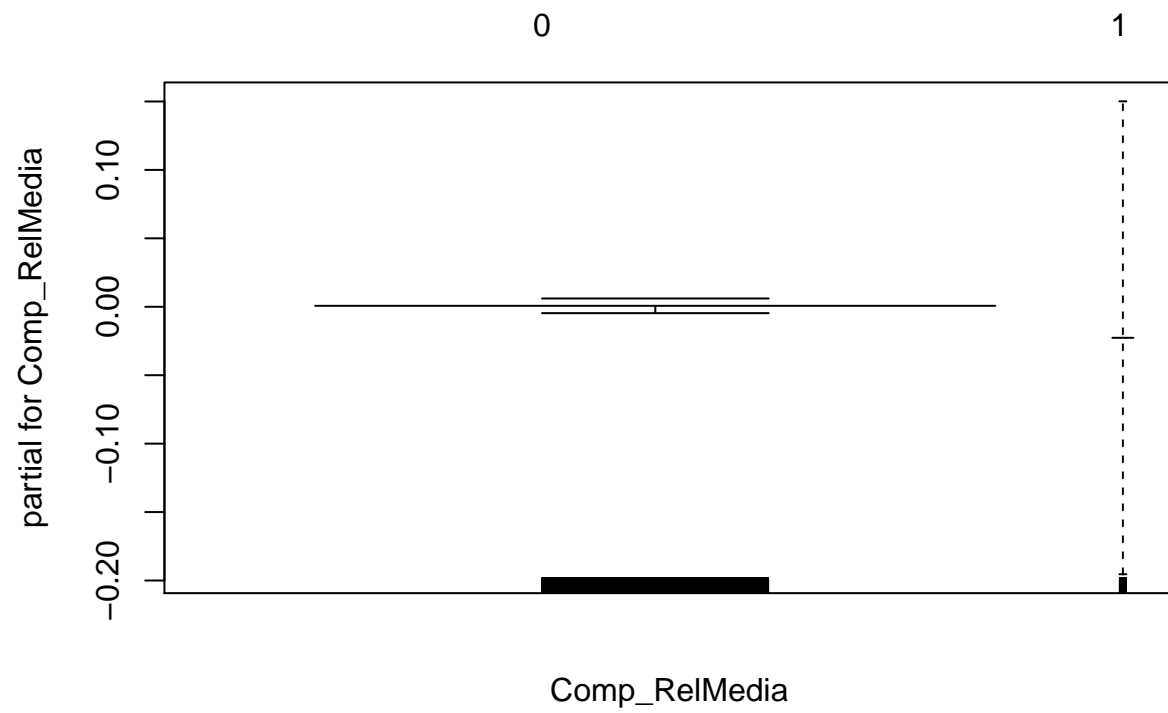


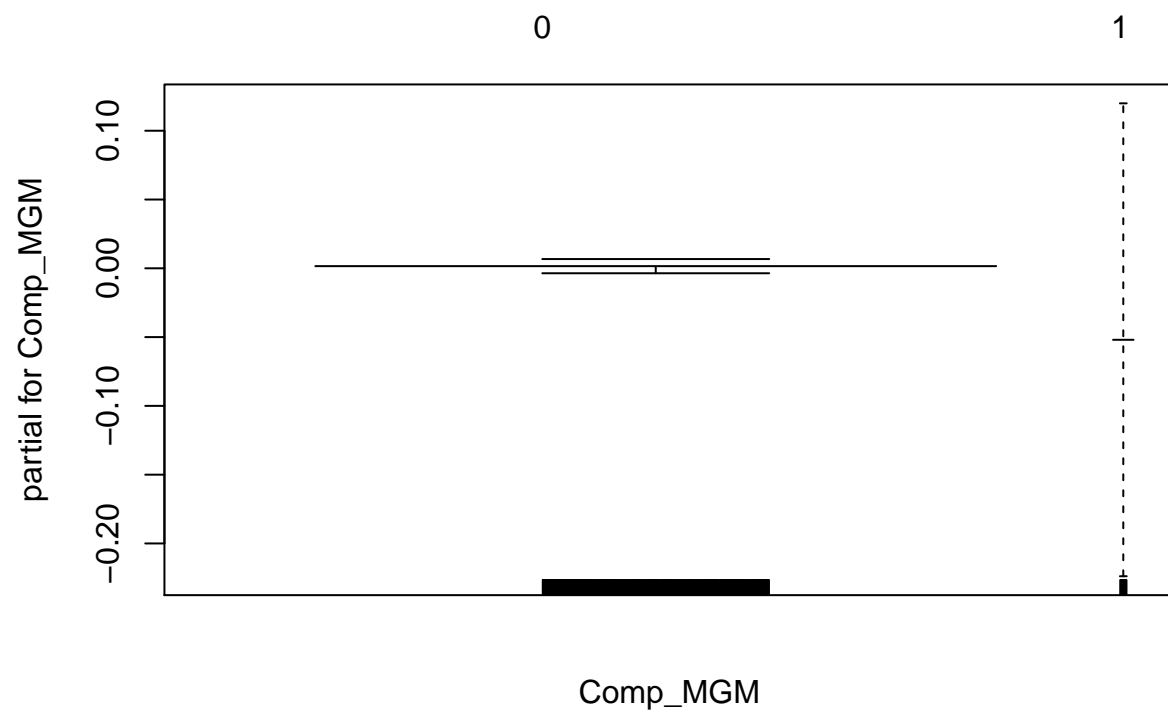


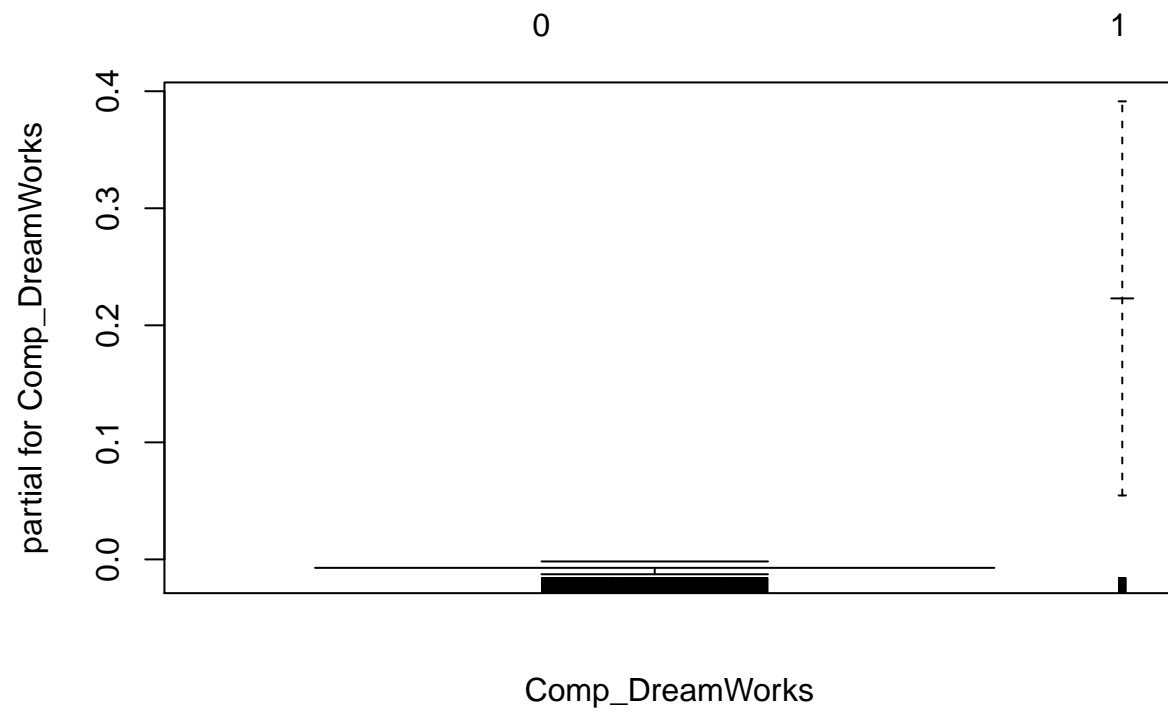


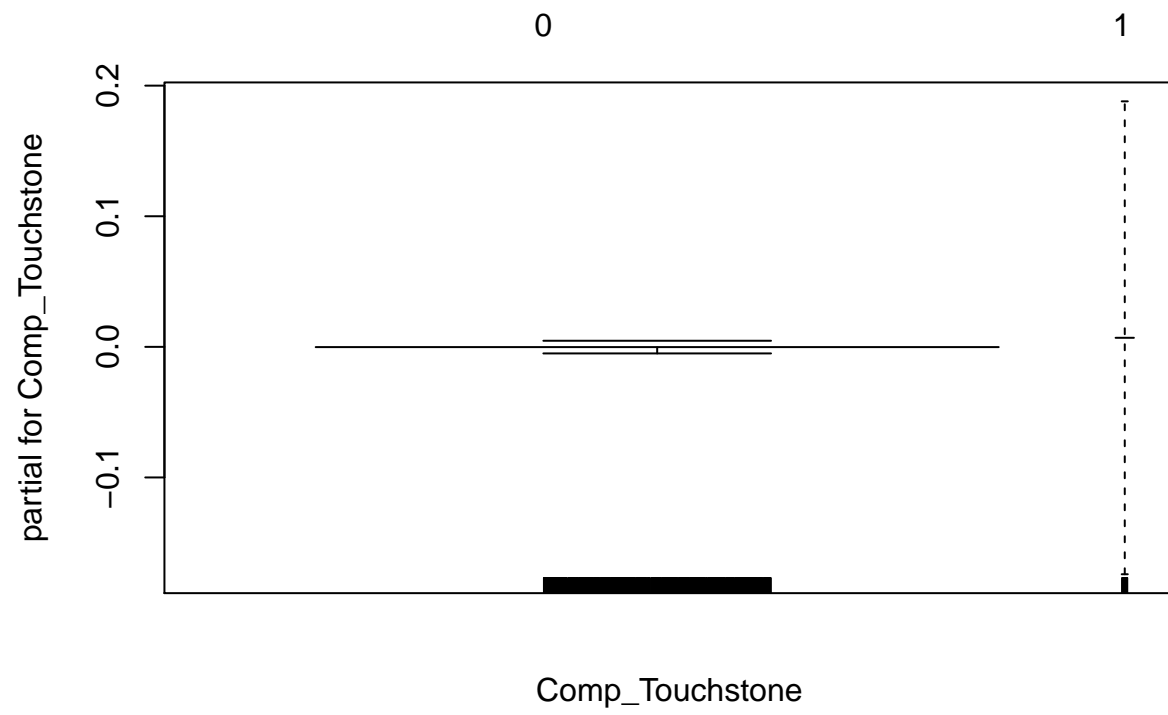


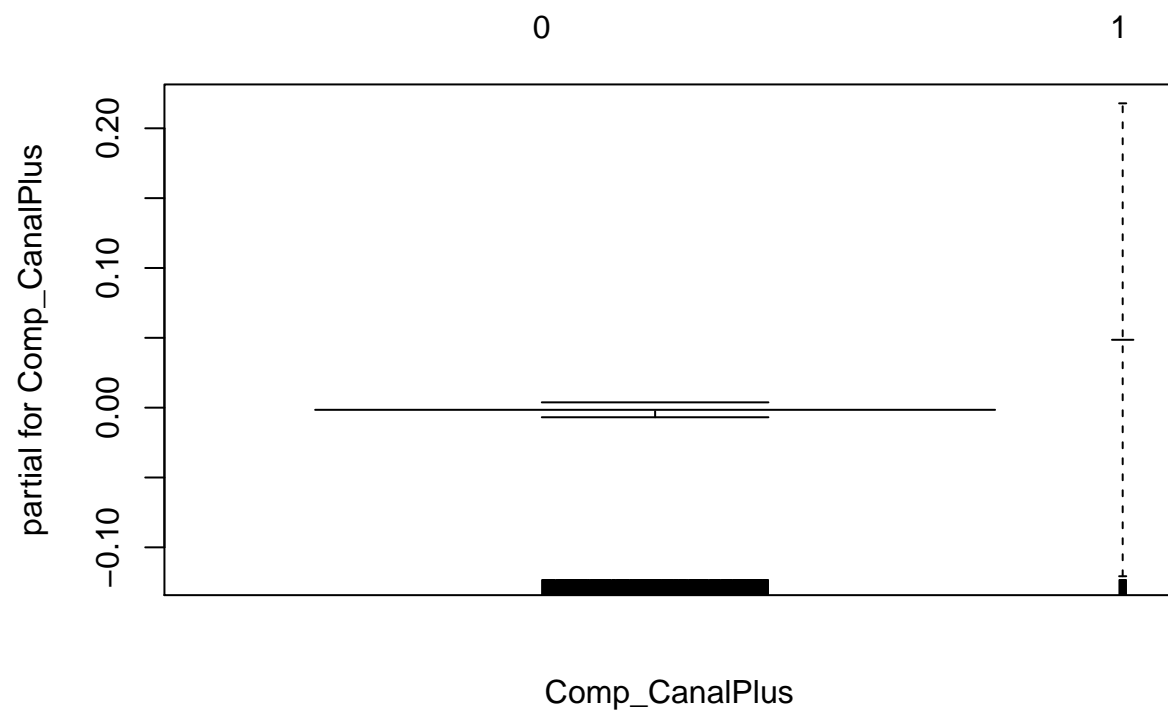


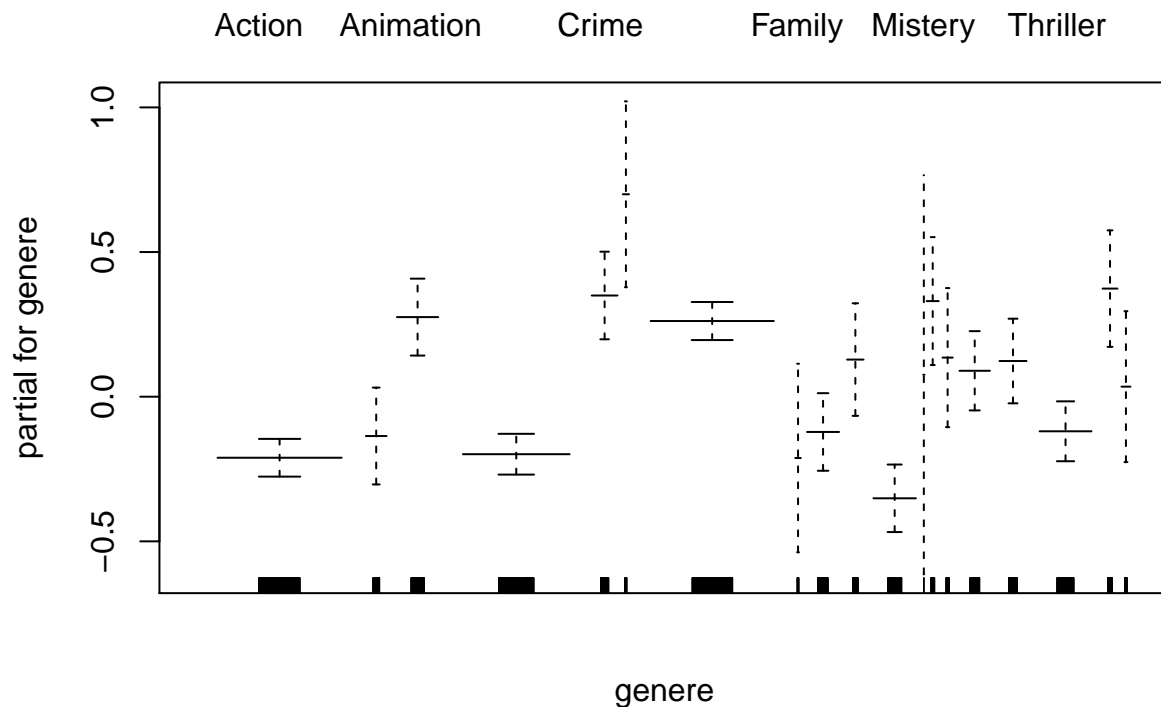












Perform stepwise selection using “gam.scope”. We specify our data, the position of our response var. and the df that we want evaluate, the function `scope()` looks for the best variables considering non-linearity with 2,3 and 4 df in this case. Values for df should be greater than 1, with df=1 implying a linear fit. The selected model is chosen considering AIC. An other important point is that this function assigns the best number of df (2,3 or 4) to those variables that are modeled with smoothing splines. `g3` is the best model found. Close to the variable we can see the total number of df, for the var. with smoothing splines we have the par. and non-par.

```
sc <- gam.scope(dati.train[, -8], response=8, arg=c("df=2", "df=3", "df=4"))
#This function doesn't return always the same model
g3 <- step.Gam(g2, scope=sc, trace=T)
```

```
## Start: vote_average ~ . - vote_classes; AIC= 4768.761
## Step:1 vote_average ~ budget + s(popularity, df = 2) + production_countries + release_date + re
## Step:2 vote_average ~ budget + s(popularity, df = 3) + production_countries + release_date + re
## Step:3 vote_average ~ s(budget, df = 2) + s(popularity, df = 3) + production_countries + releas
## Step:4 vote_average ~ s(budget, df = 2) + s(popularity, df = 3) + production_countries + releas
## Step:5 vote_average ~ s(budget, df = 2) + s(popularity, df = 4) + production_countries + releas
## Step:6 vote_average ~ s(budget, df = 3) + s(popularity, df = 4) + production_countries + releas
## Step:7 vote_average ~ s(budget, df = 3) + s(popularity, df = 4) + production_countries + s(rele
## Step:8 vote_average ~ s(budget, df = 3) + s(popularity, df = 4) + production_countries + s(rele
## Step:9 vote_average ~ s(budget, df = 3) + s(popularity, df = 4) + production_countries + s(rele
## Step:10 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(rele
## Step:11 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(rele
## Step:12 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(rele
## Step:13 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(rele
```

```
## Step:14 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:15 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:16 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:17 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:18 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:19 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:20 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:21 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:22 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:23 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:24 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:25 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:26 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
## Step:27 vote_average ~ s(budget, df = 4) + s(popularity, df = 4) + production_countries + s(release_date, df = 4) + s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages + Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train, trace = FALSE)
```

```
summary(g3)
```

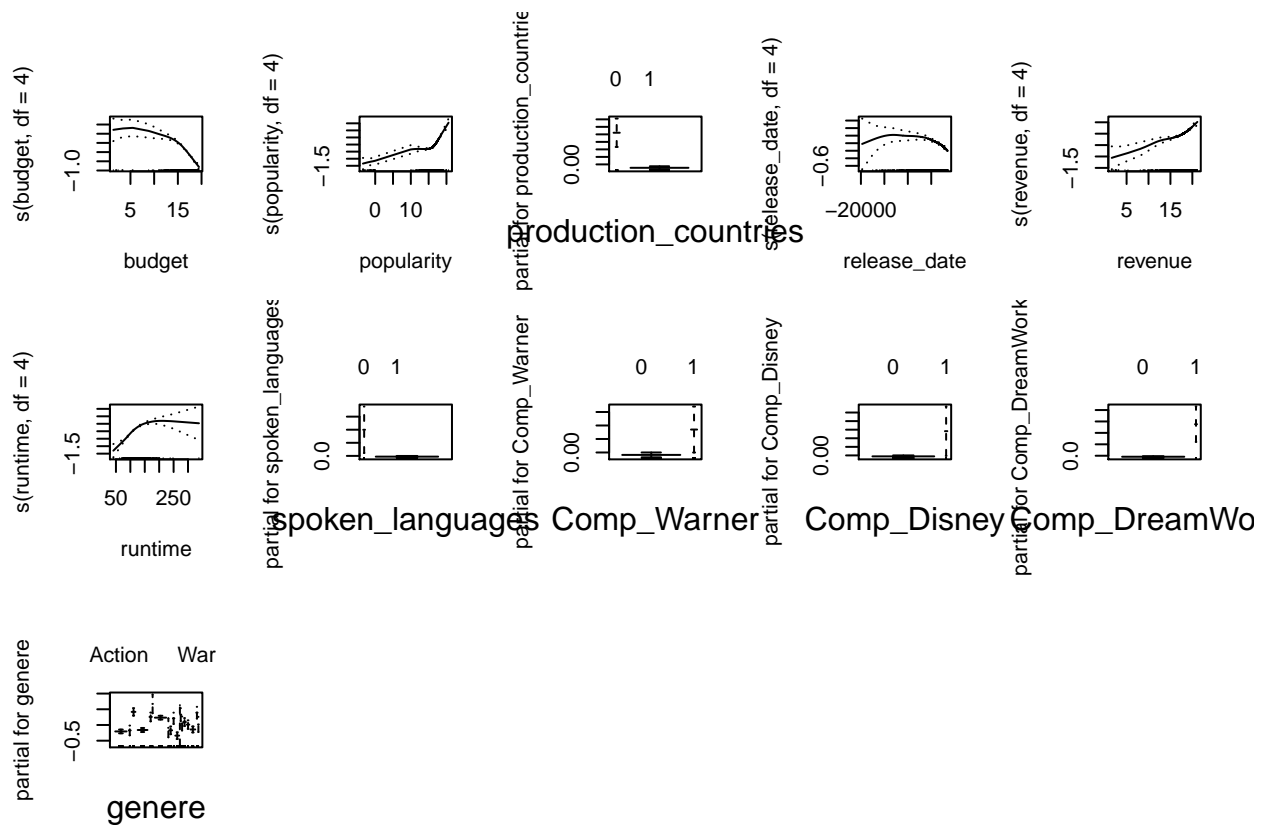
```
##
## Call: gam(formula = vote_average ~ s(budget, df = 4) + s(popularity,
##      df = 4) + production_countries + s(release_date, df = 4) +
##      s(revenue, df = 4) + s(runtime, df = 4) + spoken_languages +
##      Comp_Warner + Comp_Disney + Comp_DreamWorks + genere, data = dati.train,
##      trace = FALSE)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -5.23538 -0.35948  0.02259  0.40028  3.51055
##
## (Dispersion Parameter for gaussian family taken to be 0.4085)
##
##      Null Deviance: 1696.602 on 2259 degrees of freedom
## Residual Deviance: 905.1506 on 2216 degrees of freedom
## AIC: 4435.659
##
## Number of Local Scoring Iterations: NA
##
## Anova for Parametric Effects
##              Df Sum Sq Mean Sq  F value    Pr(>F)
## s(budget, df = 4)      1  61.03  61.028 149.4083 < 2.2e-16 ***
## s(popularity, df = 4)  1 147.97 147.968 362.2558 < 2.2e-16 ***
## production_countries   1  16.86  16.861  41.2797 1.610e-10 ***
## s(release_date, df = 4) 1  64.05  64.054 156.8172 < 2.2e-16 ***
## s(revenue, df = 4)      1  38.46  38.458  94.1523 < 2.2e-16 ***
## s(runtime, df = 4)      1 189.85 189.849 464.7908 < 2.2e-16 ***
## spoken_languages       1   1.47   1.466   3.5898 0.058267 .
## Comp_Warner            1   0.75   0.752   1.8402 0.175061
## Comp_Disney            1   4.31   4.314  10.5626 0.001171 **
## Comp_DreamWorks        1  12.73  12.727  31.1583 2.669e-08 ***
## genere                 18 115.34   6.408  15.6873 < 2.2e-16 ***
## Residuals             2216 905.15   0.408
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##              Npar Df Npar F      Pr(F)
```

```
## (Intercept)
## s(budget, df = 4)          3 35.280 < 2.2e-16 ***
## s(popularity, df = 4)     3 71.661 < 2.2e-16 ***
## production_countries
## s(release_date, df = 4)   3  6.294 0.0002999 ***
## s(revenue, df = 4)        3  6.285 0.0003035 ***
## s(runtime, df = 4)        3 21.216 1.503e-13 ***
## spoken_languages
## Comp_Warner
## Comp_Disney
## Comp_DreamWorks
## genere
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
AIC(g3)
```

```
## [1] 4435.659
```

```
par(mfrow=c(3,5))
plot(g3, se=T)
par(mfrow=c(1,1))
```



```
#If we want to see better some plot
#plot(g3, se=T, ask=T)
```

The GAM model is better than the linear model in terms of predictions

```
#Prediction
dati.test[,c(3,7, 10:24)]= lapply(dati.test[,c(3,7, 10:24)],factor)
p.gam <- predict(g3, newdata=dati.test)
#The deviance is lower than linear model. The GAM predict in a better way and is also more
#interpretable
dev.gam <- sum((p.gam-dati.test$vote_average)^2)
dev.gam
```

```
## [1] 442.8381
```

We compare the linear model m2 (obtained by stepwise selection) and g3 (considering the AIC). We do NOT compare m1 and g3, m1 is the starting point in the selection procedure of g3, so we definitely find a better model.

Gradient Boosting

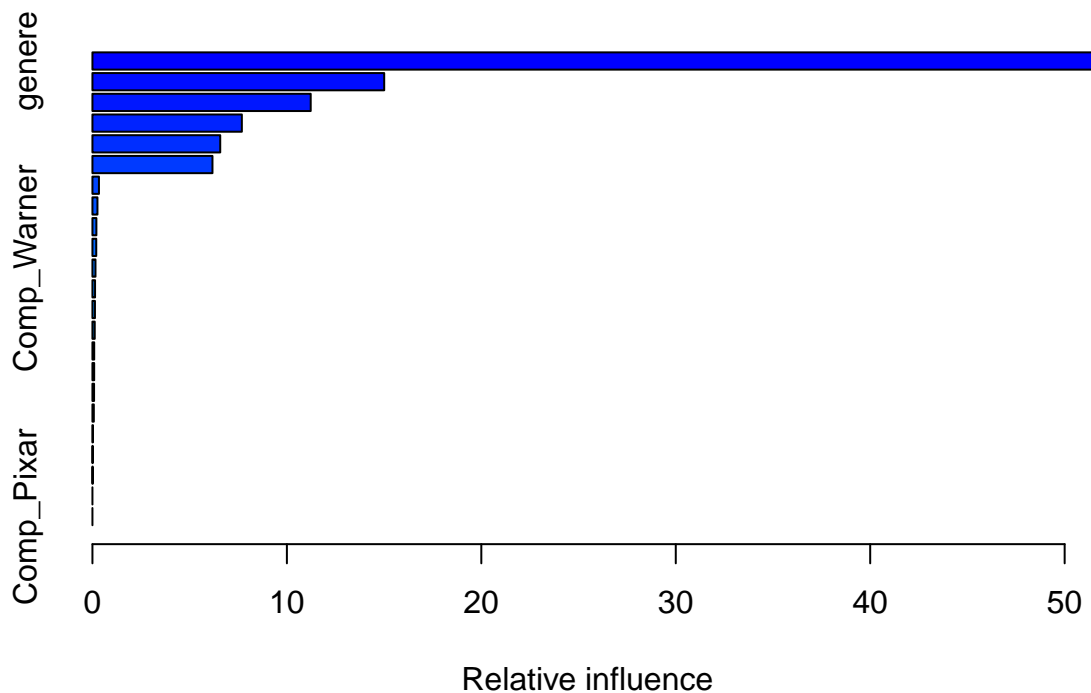
```
library (gbm)
```

```
## Loaded gbm 2.1.8
```

```
#Divide the training set into 2 parts (70 and 30) to select the best number of trees
#depth (with or without interactions) and the best number of trees (out two hyperparameters)
set.seed(999)
tt = sample (1:nrow(dati.train), 0.7*nrow(dati.train))
train.1=dati.train[tt ,]
train.2=dati.train[-tt ,]
```

Model with all the variables except “vote_classes” (highly correlated with our response); now we have a numerical response, with a categorical one we have to change the distribution. First boosting for regression (stump: we do not allow for interactions between variables, depth=1)

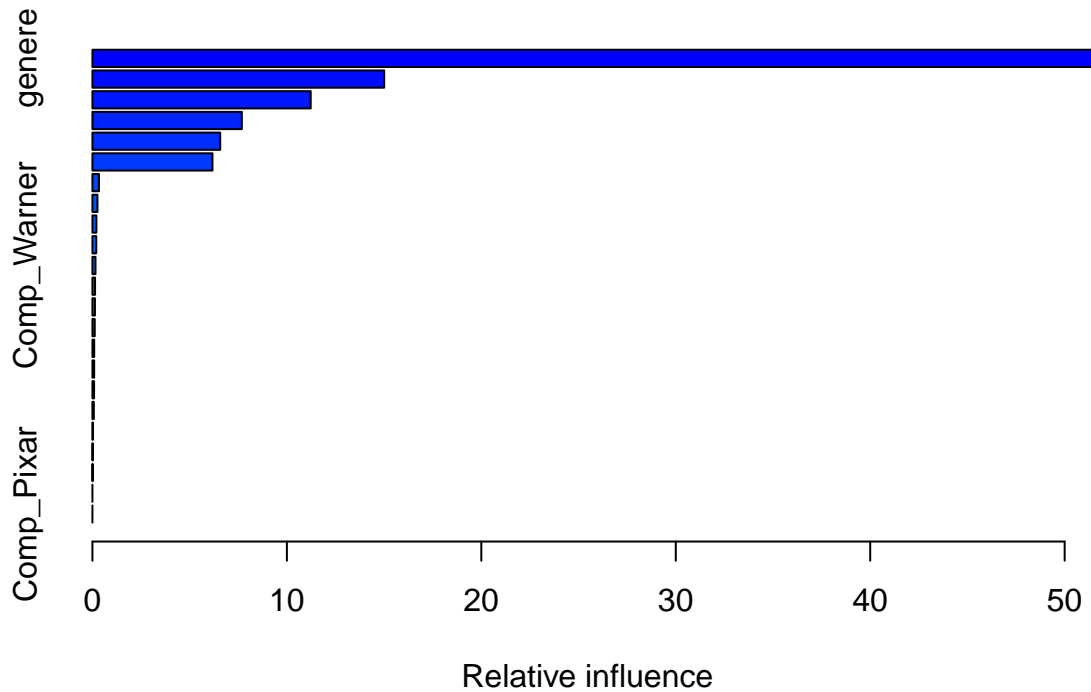
```
boost.movies=gbm(vote_average ~ .-vote_classes, data=train.1,
                 distribution="gaussian", n.trees=5000, interaction.depth=1)
summary(boost.movies)
```



```
##                var      rel.inf
## genere          genere 51.42479266
## popularity      popularity 15.00551819
## revenue         revenue 11.22495309
## runtime         runtime  7.68403156
## release_date    release_date 6.57186399
## budget          budget  6.17328368
## spoken_languages spoken_languages 0.33480196
## Comp_20fox      Comp_20fox 0.25842739
## production_countries production_countries 0.20239605
## Comp_Columbia   Comp_Columbia 0.19606125
## Comp_Warner     Comp_Warner 0.15630788
## Comp_Touchstone Comp_Touchstone 0.13455470
## Comp_Paramount  Comp_Paramount 0.12926298
## Comp_DreamWorks Comp_DreamWorks 0.11765897
## Comp_Universal  Comp_Universal 0.09118150
## Comp_CanalPlus  Comp_CanalPlus 0.08924636
## Comp_Disney     Comp_Disney 0.08297461
## Comp_Miramax    Comp_Miramax 0.07093295
## Comp_NewLine    Comp_NewLine 0.02420903
## Comp_RelMedia   Comp_RelMedia 0.01536959
## Comp_MGM        Comp_MGM 0.01217161
## Comp_VillageRoadshow Comp_VillageRoadshow 0.00000000
## Comp_Pixar      Comp_Pixar 0.00000000
```

Second boosting: increase the depth of trees (4)


```
boost.movies.1=gbm(vote_average ~ .-vote_classes, data=train.1,
                    distribution="gaussian", n.trees=5000, interaction.depth=4)
summary(boost.movies)
```



```
##               var      rel.inf
## genere         genere 51.42479266
## popularity     popularity 15.00551819
## revenue        revenue 11.22495309
## runtime        runtime  7.68403156
## release_date   release_date 6.57186399
## budget         budget  6.17328368
## spoken_languages spoken_languages 0.33480196
## Comp_20fox      Comp_20fox 0.25842739
## production_countries production_countries 0.20239605
## Comp_Columbia   Comp_Columbia 0.19606125
## Comp_Warner     Comp_Warner 0.15630788
## Comp_Touchstone Comp_Touchstone 0.13455470
## Comp_Paramount  Comp_Paramount 0.12926298
## Comp_DreamWorks Comp_DreamWorks 0.11765897
## Comp_Universal  Comp_Universal 0.09118150
## Comp_CanalPlus  Comp_CanalPlus 0.08924636
## Comp_Disney     Comp_Disney 0.08297461
## Comp_Miramax    Comp_Miramax 0.07093295
## Comp_NewLine    Comp_NewLine 0.02420903
## Comp_RelMedia   Comp_RelMedia 0.01536959
```

```
## Comp_MGM                      Comp_MGM  0.01217161
## Comp_VillageRoadshow Comp_VillageRoadshow  0.00000000
## Comp_Pixar                    Comp_Pixar  0.00000000
```

```
#Predictions for our two model, to select the best one
#yhat.boost: matrix with 678 rows (samples) and 5000 columns (trees)
yhat.boost=predict(boost.movies, newdata=train.2, n.trees=1:5000)
yhat.boost.1=predict(boost.movies.1, newdata=train.2, n.trees=1:5000)
```

Calculate the error for each iteration (5000). Use ‘apply’ to perform a ‘cycle for’, then the first element is the matrix we want to use, the “2” means ‘by column’ and the third element indicates

```
#the function we want to calculate
err = apply(yhat.boost, 2, function(pred) mean((train.2$vote_average - pred)^2))
err.1 = apply(yhat.boost.1, 2, function(pred) mean((train.2$vote_average - pred)^2))
```

```
#Best error for each model
best0=which.min(err)
min(err)
```

```
## [1] 0.4671261
```

```
#The second model has a minor prediction error
best1=which.min(err.1)
min(err.1)
```

```
## [1] 0.4505591
```

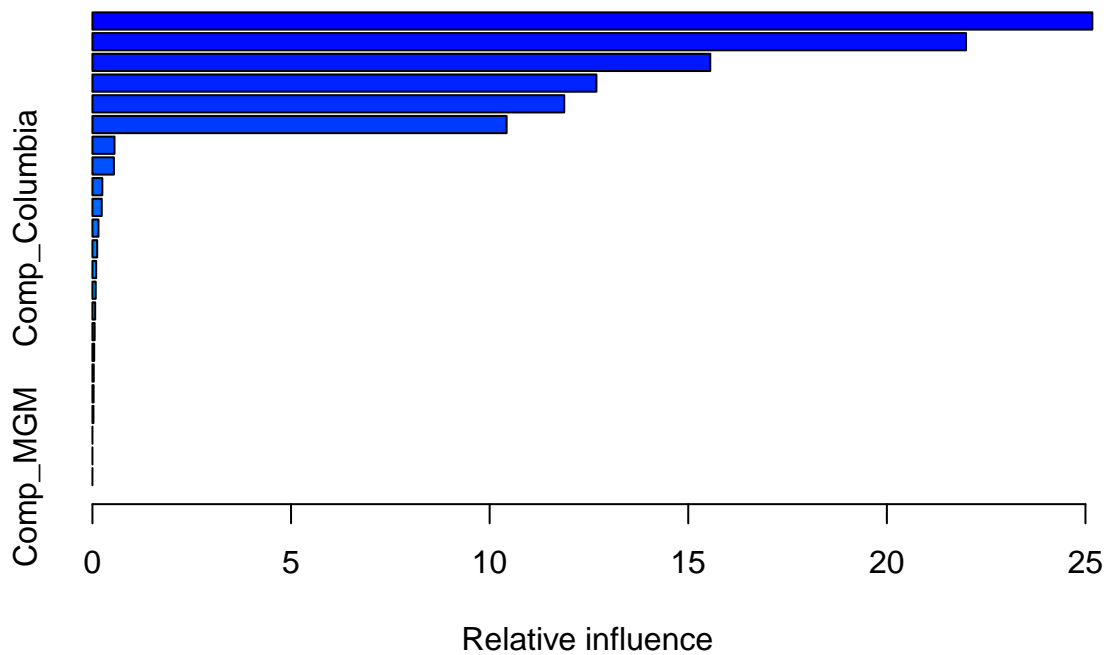
```
#Since min(err.1) is smaller than min(err), depth=4 is better than depth=1, we select best1
best1
```

```
## 313
## 313
```

```
best<- best1
```

Final model with best number of trees on entire training set (dati.train)

```
boost.movies.1=gbm(vote_average ~ .-vote_classes, data=dati.train,
                    distribution="gaussian", n.trees=best, interaction.depth=4)
summary(boost.movies.1)
```



```
##          var      rel.inf
## popularity      popularity 25.17359710
## genere          genere 21.99307840
## runtime         runtime 15.55475416
## revenue         revenue 12.68825133
## budget          budget 11.87818316
## release_date    release_date 10.42741231
## production_countries production_countries 0.55613258
## spoken_languages spoken_languages 0.54020424
## Comp_20fox      Comp_20fox 0.24896541
## Comp_DreamWorks Comp_DreamWorks 0.23416743
## Comp_Columbia   Comp_Columbia 0.15205890
## Comp_CanalPlus  Comp_CanalPlus 0.11929316
## Comp_Disney     Comp_Disney 0.09418756
## Comp_Warner     Comp_Warner 0.08266237
## Comp_Miramax    Comp_Miramax 0.06885655
## Comp_Universal  Comp_Universal 0.05555052
## Comp_RelMedia   Comp_RelMedia 0.04431503
## Comp_Touchstone Comp_Touchstone 0.03322879
## Comp_Paramount  Comp_Paramount 0.02970096
## Comp_NewLine    Comp_NewLine 0.02540003
## Comp_VillageRoadshow Comp_VillageRoadshow 0.00000000
## Comp_Pixar      Comp_Pixar 0.00000000
## Comp_MGM        Comp_MGM 0.00000000
```

```
# AIC(boost.movies.1)
```

```
#Prediction on test set
```

```
p.boost=predict(boost.movies.1, newdata=dati.test, n.trees=best)
```

```
dev.boost <- sum((p.boost-dati.test$vote_average)^2)
```

```
#To compare with linear model and GAM. GBM is the best model considering deviance.
```

```
dev.boost
```

```
## [1] 423.6082
```

Change the plot to improve readability. Increase space on the left to fit the name of variables.

```
#Default parameters
```

```
mai.old<-par()$mai
```

```
mai.old
```

```
## [1] 1.02 0.82 0.82 0.42
```

```
#New parameters equal to old parameters
```

```
mai.new<-mai.old
```

```
#Substitute parameter relative to left space
```

```
mai.new[2] <- 2.1
```

```
mai.new
```

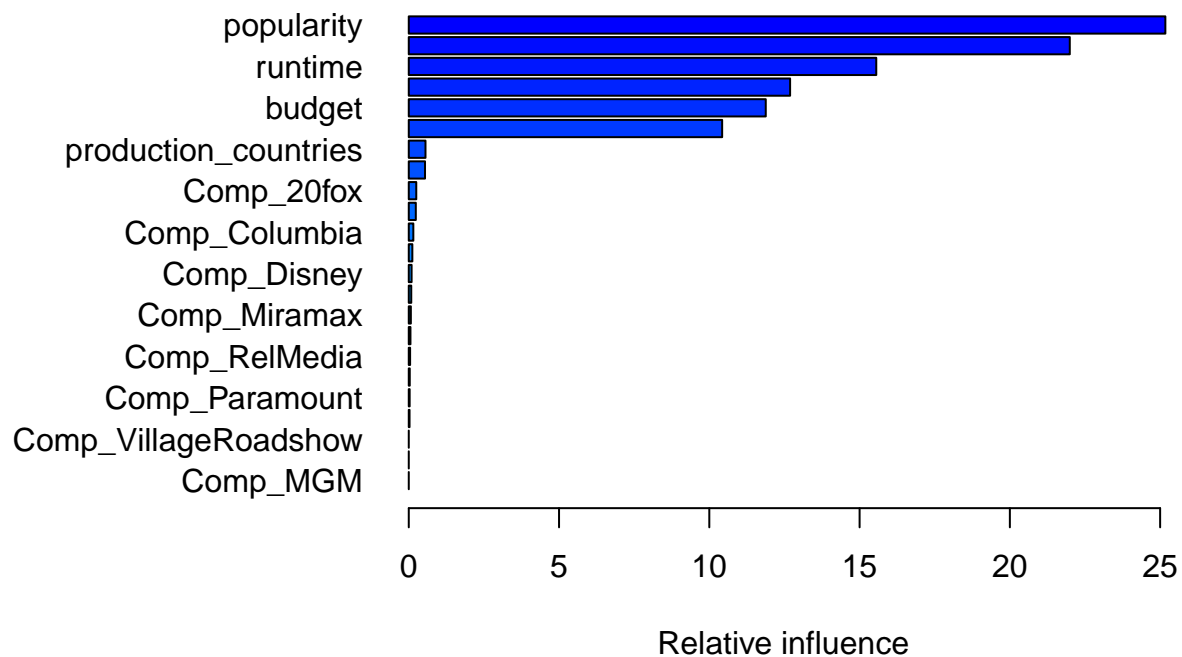
```
## [1] 1.02 2.10 0.82 0.42
```

```
#Modify graphical parameters
```

```
par(mai=mai.new)
```

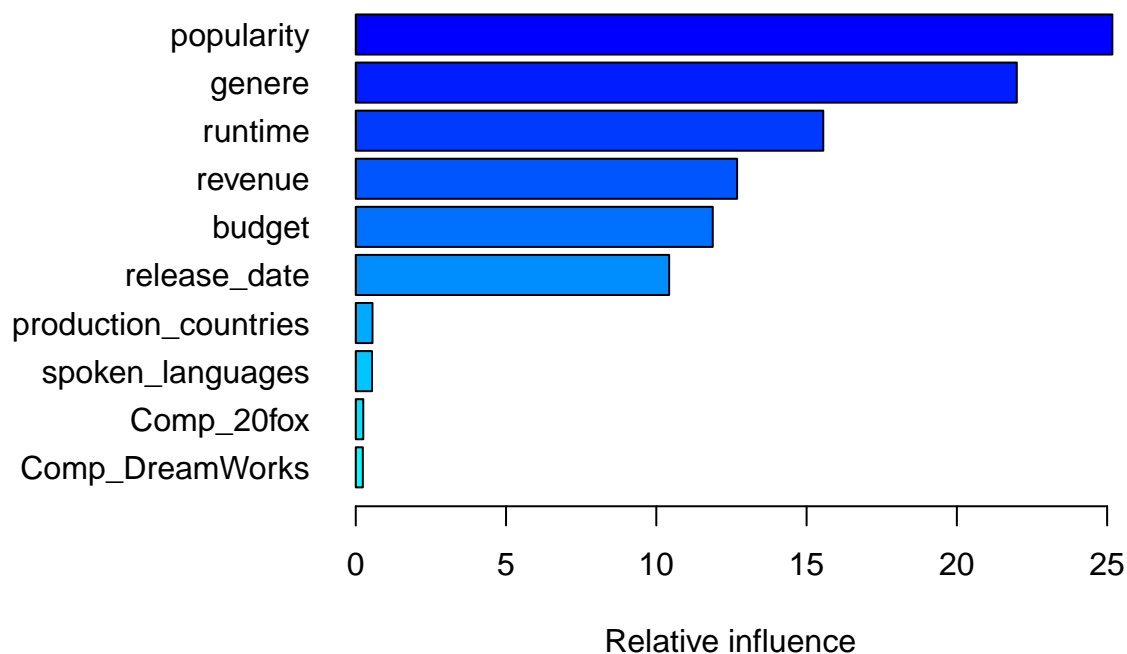
```
#las=1 horizontal names on y
```

```
summary(boost.movies.1, las=1)
```



##		var	rel.inf
##	popularity	popularity	25.17359710
##	genre	genre	21.99307840
##	runtime	runtime	15.55475416
##	revenue	revenue	12.68825133
##	budget	budget	11.87818316
##	release_date	release_date	10.42741231
##	production_countries	production_countries	0.55613258
##	spoken_languages	spoken_languages	0.54020424
##	Comp_20fox	Comp_20fox	0.24896541
##	Comp_DreamWorks	Comp_DreamWorks	0.23416743
##	Comp_Columbia	Comp_Columbia	0.15205890
##	Comp_CanalPlus	Comp_CanalPlus	0.11929316
##	Comp_Disney	Comp_Disney	0.09418756
##	Comp_Warner	Comp_Warner	0.08266237
##	Comp_Miramax	Comp_Miramax	0.06885655
##	Comp_Universal	Comp_Universal	0.05555052
##	Comp_RelMedia	Comp_RelMedia	0.04431503
##	Comp_Touchstone	Comp_Touchstone	0.03322879
##	Comp_Paramount	Comp_Paramount	0.02970096
##	Comp_NewLine	Comp_NewLine	0.02540003
##	Comp_VillageRoadshow	Comp_VillageRoadshow	0.00000000
##	Comp_Pixar	Comp_Pixar	0.00000000
##	Comp_MGM	Comp_MGM	0.00000000

```
#cBar sets how many variables to draw
summary(boost.movies.1, las=1, cBar=10)
```



```
##           var      rel.inf
## popularity      popularity 25.17359710
## genere          genere    21.99307840
## runtime         runtime   15.55475416
## revenue         revenue   12.68825133
## budget          budget    11.87818316
## release_date    release_date 10.42741231
## production_countries production_countries 0.55613258
## spoken_languages spoken_languages 0.54020424
## Comp_20fox      Comp_20fox  0.24896541
## Comp_DreamWorks Comp_DreamWorks 0.23416743
## Comp_Columbia   Comp_Columbia 0.15205890
## Comp_CanalPlus  Comp_CanalPlus 0.11929316
## Comp_Disney     Comp_Disney   0.09418756
## Comp_Warner     Comp_Warner   0.08266237
## Comp_Miramax    Comp_Miramax  0.06885655
## Comp_Universal  Comp_Universal 0.05555052
## Comp_RelMedia   Comp_RelMedia 0.04431503
## Comp_Touchstone Comp_Touchstone 0.03322879
## Comp_Paramount  Comp_Paramount 0.02970096
## Comp_NewLine    Comp_NewLine  0.02540003
## Comp_VillageRoadshow Comp_VillageRoadshow 0.00000000
```

```
## Comp_Pixar          Comp_Pixar  0.00000000
## Comp_MGM            Comp_MGM   0.00000000
```

```
#Back to old parameters
```

```
par(mai=mai.old)
```

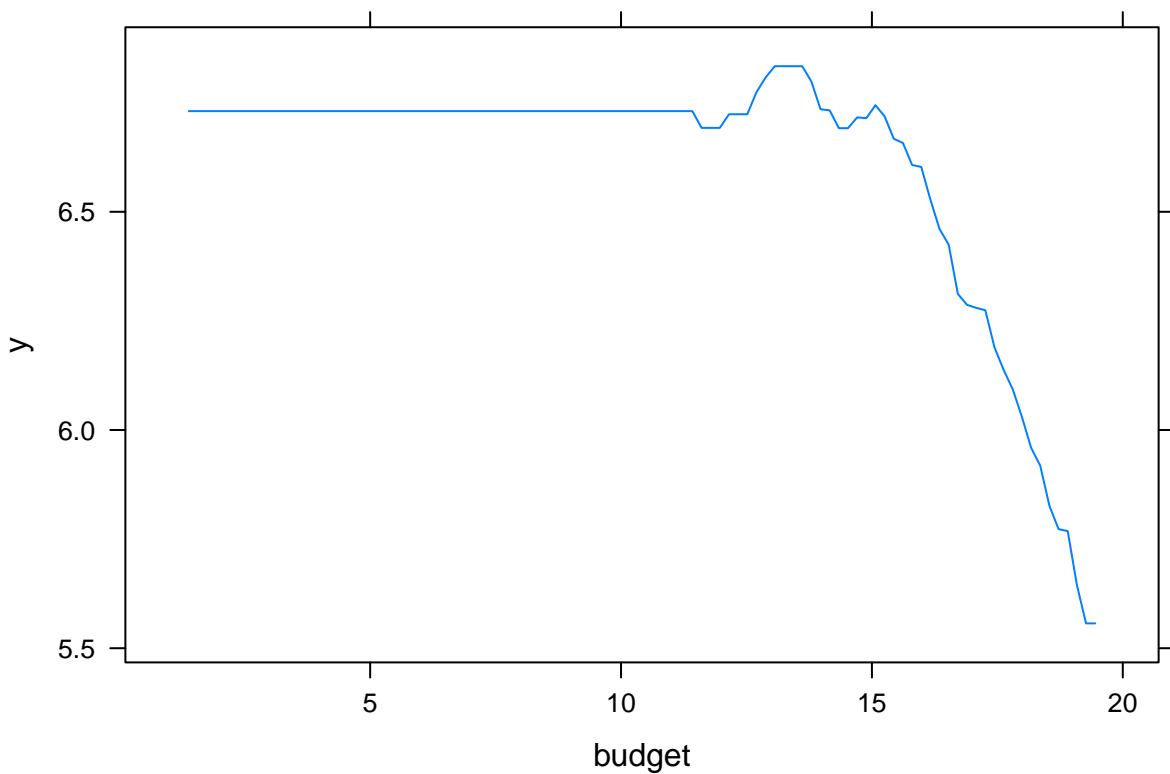
This plot doesn't say anything about the nature or intensity of the relationship with the response variable, it is just about the relative importance, in term of reduction of error any time we perform an iteration within our model

```
#Partial dependence plots
```

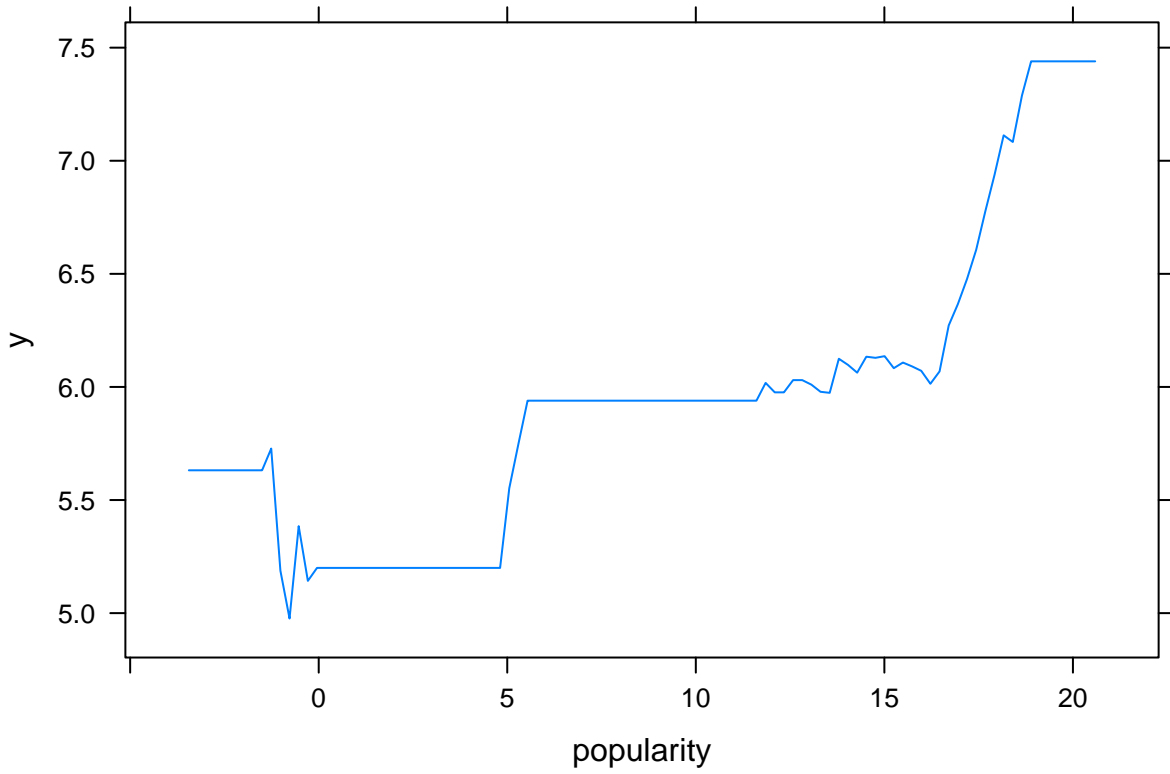
```
#Univariate
```

```
#Budget: we see the same relation also in linear and GAM model, this is a consistent result
```

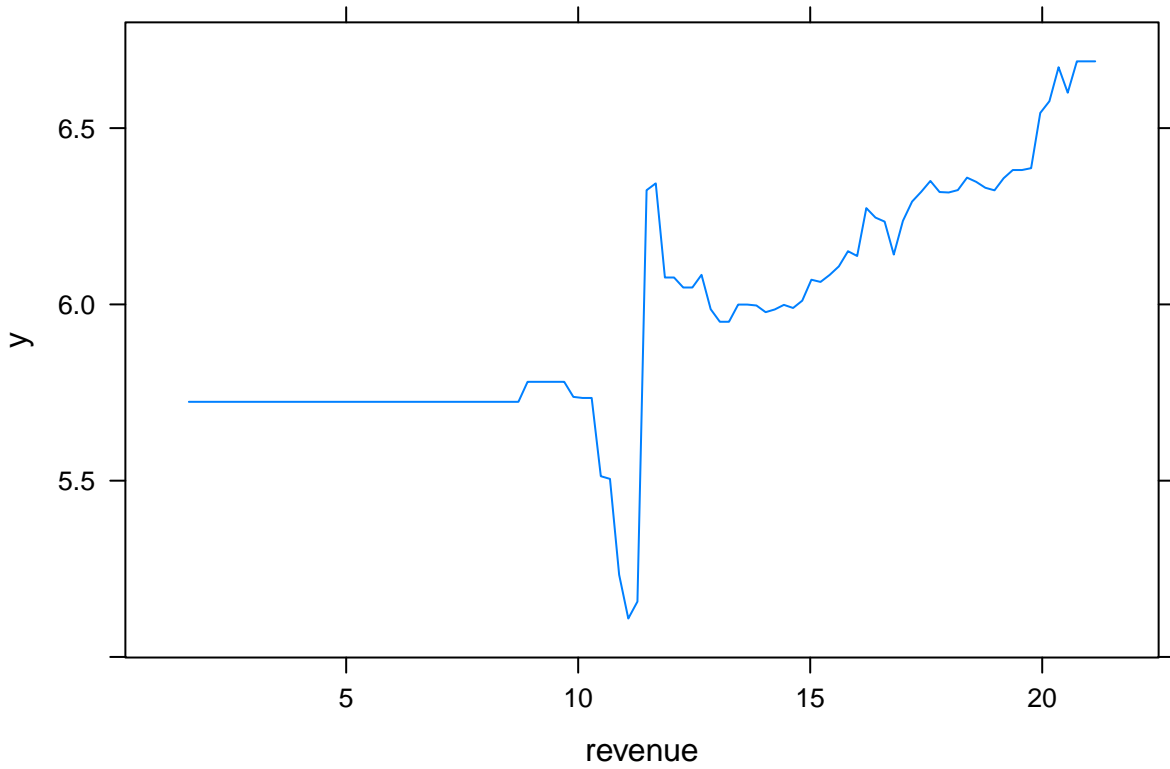
```
plot(boost.movies.1, i.var=1, n.trees = best)
```



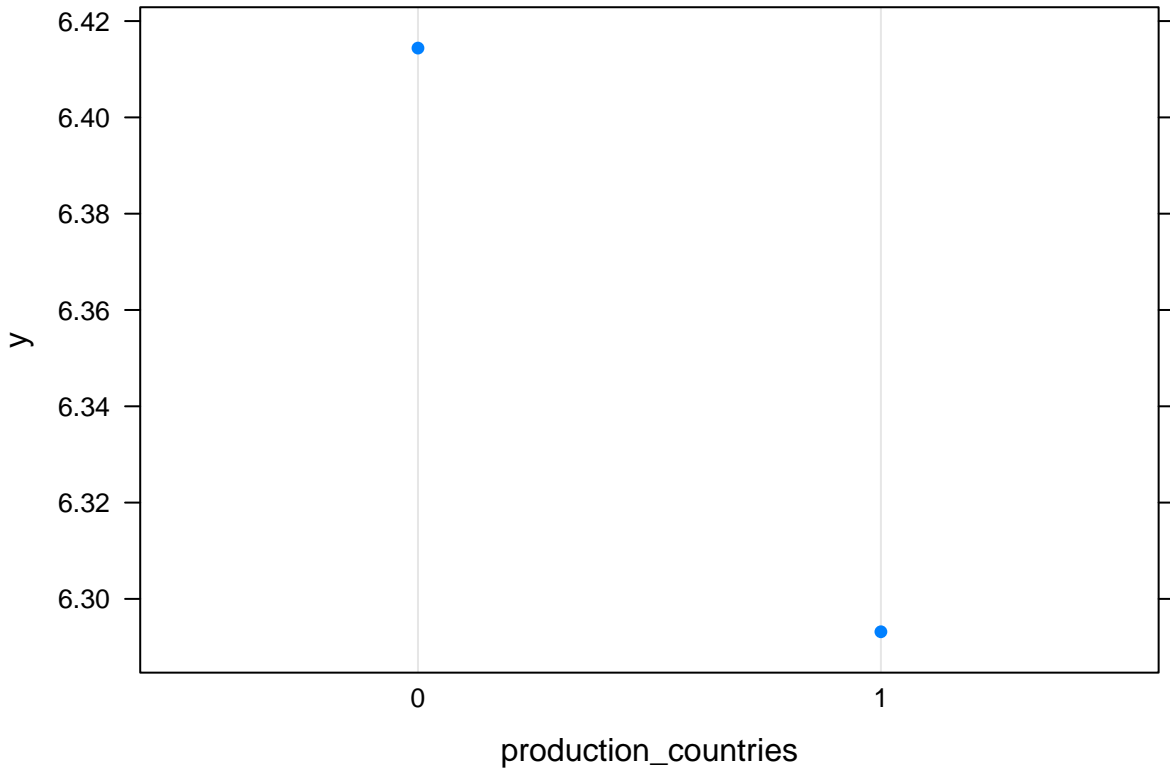
```
plot(boost.movies.1, i.var=2, n.trees = best)
```



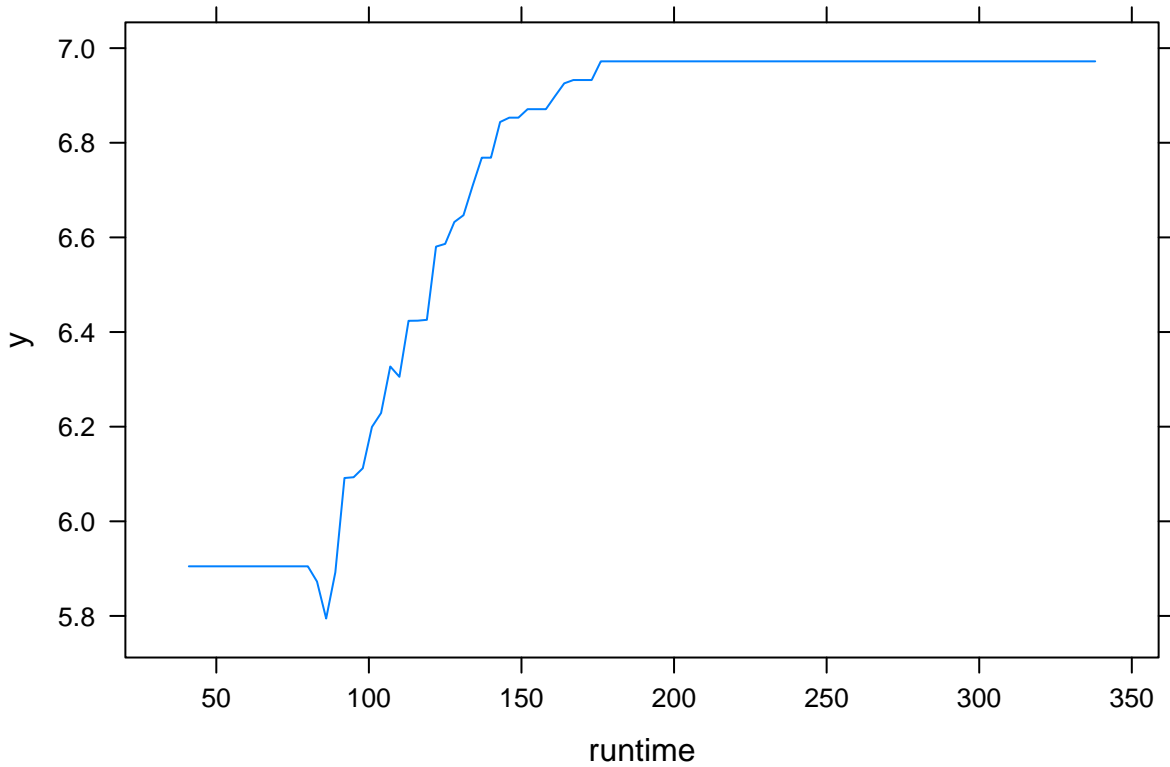
```
plot(boost.movies.1, i.var=5, n.trees = best)
```

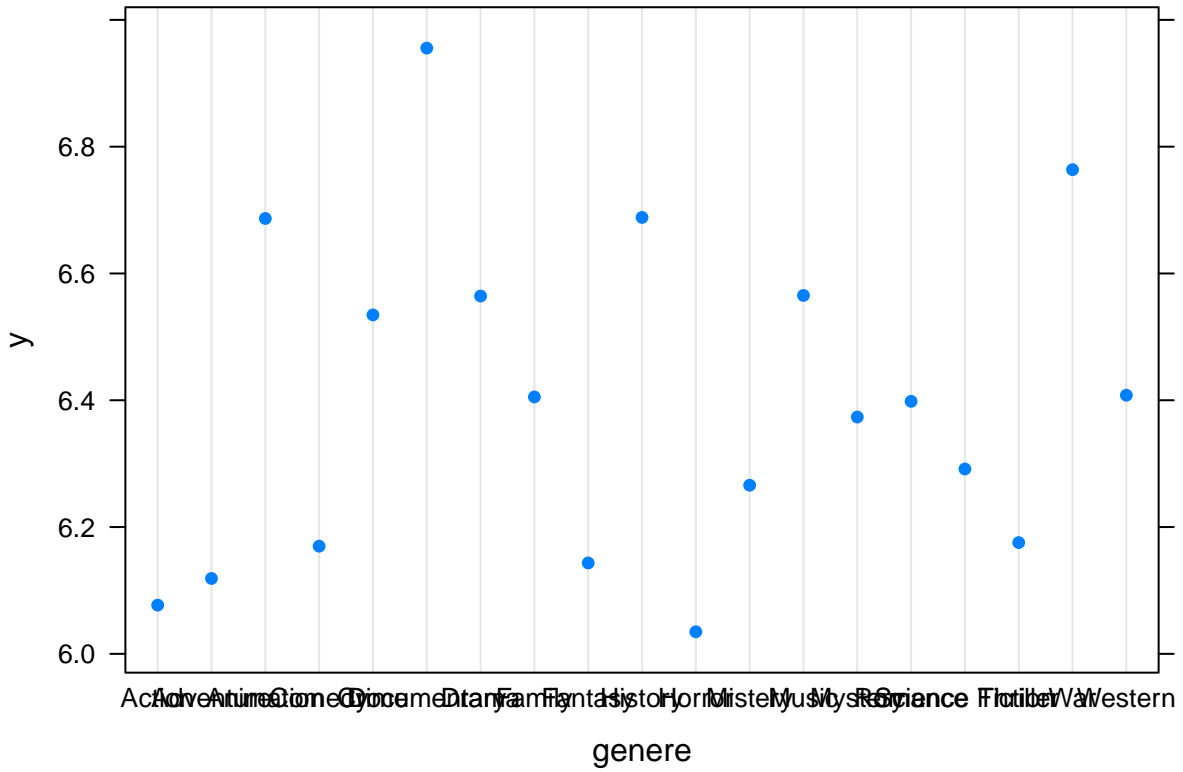
```
plot(boost.movies.1, i.var=3, n.trees = best)
```



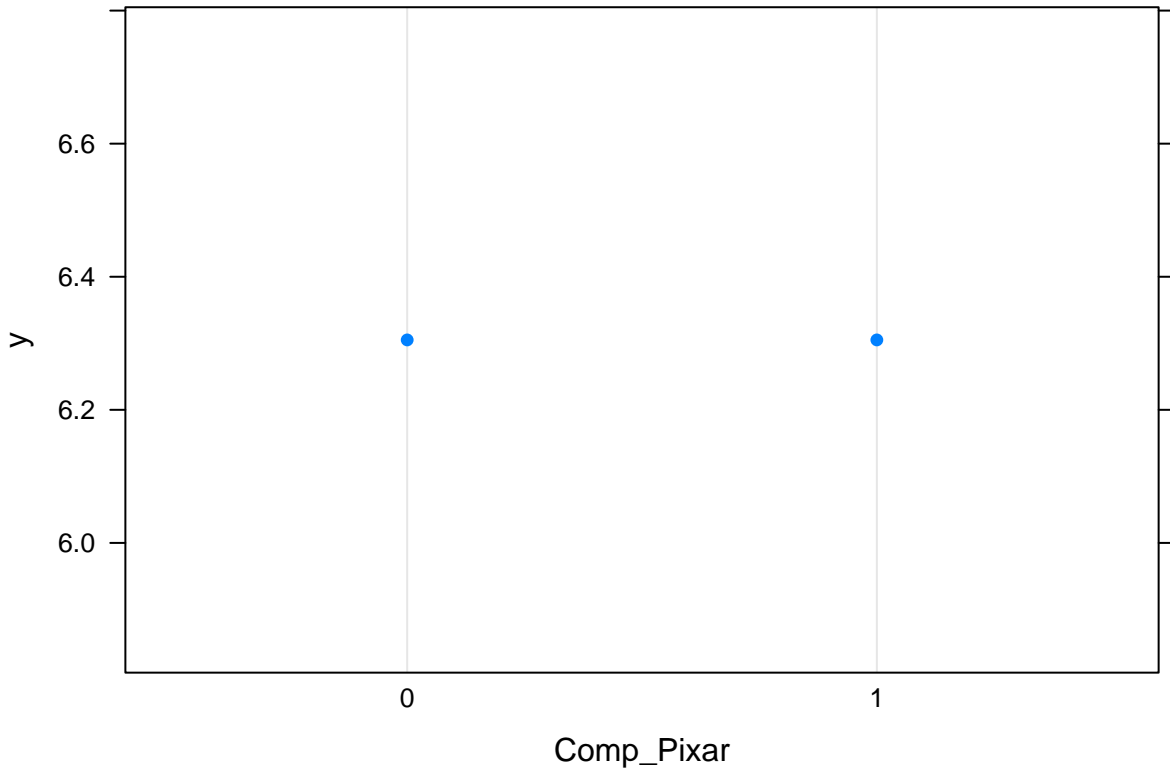
```
plot(boost.movies.1, i.var=6, n.trees = best)
```



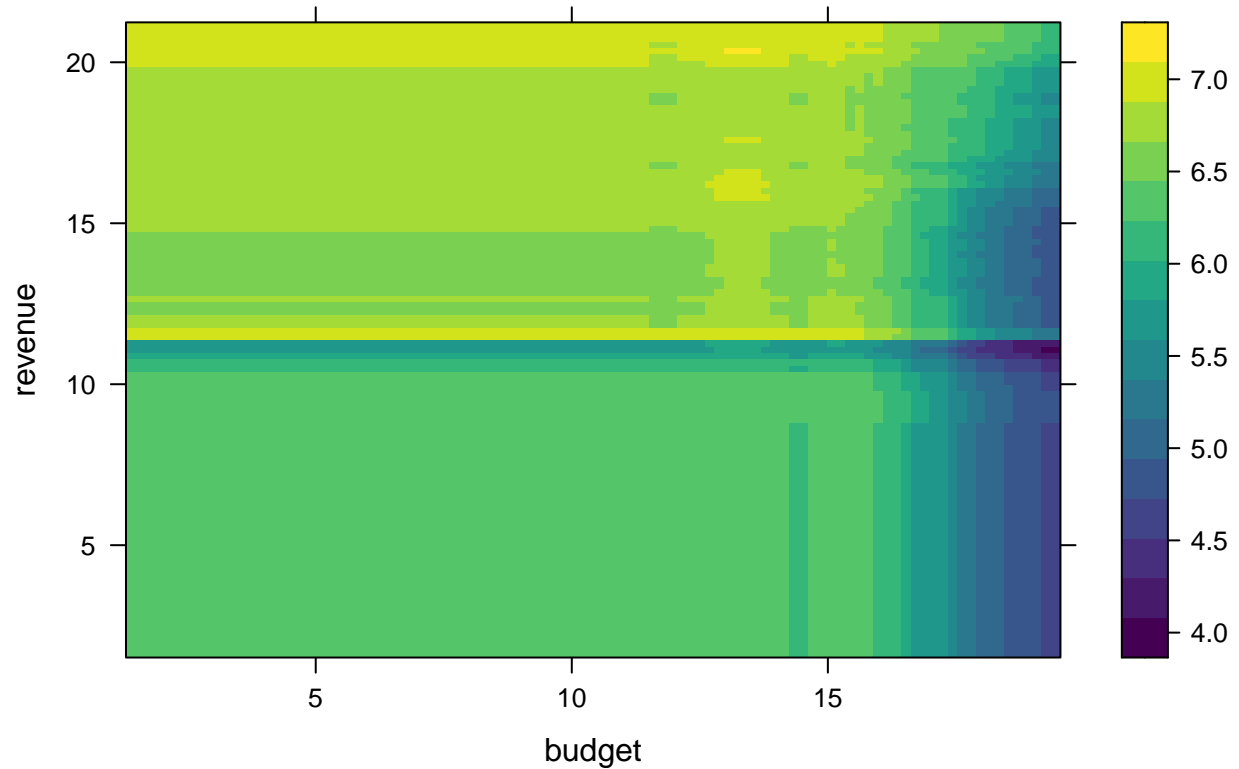
```
#Qualitative variable  
plot(boost.movies.1, i=23, n.trees = best)
```



```
#No effect
plot(boost.movies.1, i=17, n.trees = best)
```



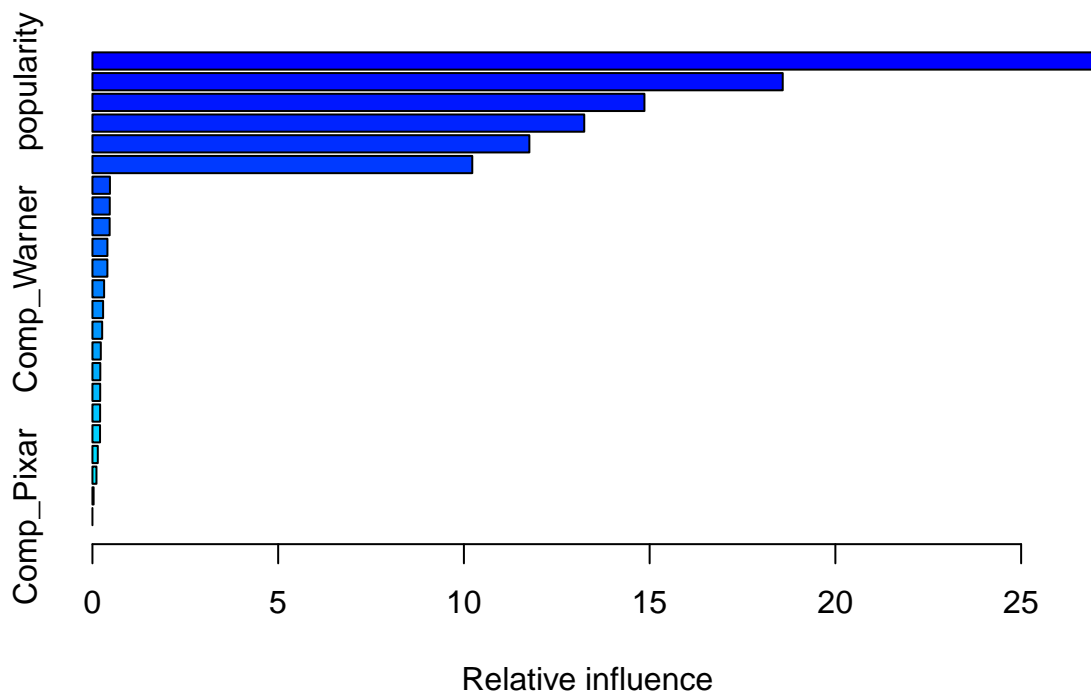
```
#Bivariate  
plot(boost.movies.1, i.var=c(1,5), n.trees = best)
```



*#REMEMBER: a PDP indicates what is the relation but after having accounted for the presence of
all the variables within the model, very different from the GAM models: we have a pure
relationship (net relationship)*

Exercise: try to fit another GB with shrinkage=0.1

```
#Model
boost.movies.2=gbm(vote_average ~ .-vote_classes ,data=train.1,
                    distribution="gaussian",n.trees=5000, interaction.depth=4, shrinkage=0.1)
summary(boost.movies.2)
```



##	var	rel.inf
## genere	genere	26.91773096
## popularity	popularity	18.58232511
## revenue	revenue	14.85981433
## release_date	release_date	13.23954222
## runtime	runtime	11.76211677
## budget	budget	10.22536421
## Comp_Columbia	Comp_Columbia	0.47078154
## Comp_Touchstone	Comp_Touchstone	0.46690457
## production_countries	production_countries	0.46382649
## spoken_languages	spoken_languages	0.40319078
## Comp_20fox	Comp_20fox	0.40271238
## Comp_Warner	Comp_Warner	0.31443782
## Comp_CanalPlus	Comp_CanalPlus	0.28732629
## Comp_Universal	Comp_Universal	0.26229142
## Comp_RelMedia	Comp_RelMedia	0.22549716
## Comp_DreamWorks	Comp_DreamWorks	0.21184481
## Comp_NewLine	Comp_NewLine	0.20865814
## Comp_Paramount	Comp_Paramount	0.20642503
## Comp_Disney	Comp_Disney	0.20432901
## Comp_Miramax	Comp_Miramax	0.14453861
## Comp_MGM	Comp_MGM	0.10873545
## Comp_VillageRoadshow	Comp_VillageRoadshow	0.03160691
## Comp_Pixar	Comp_Pixar	0.00000000

```

#Predictions
yhat.boost.2=predict(boost.movies, newdata=train.2, n.trees=1:5000)

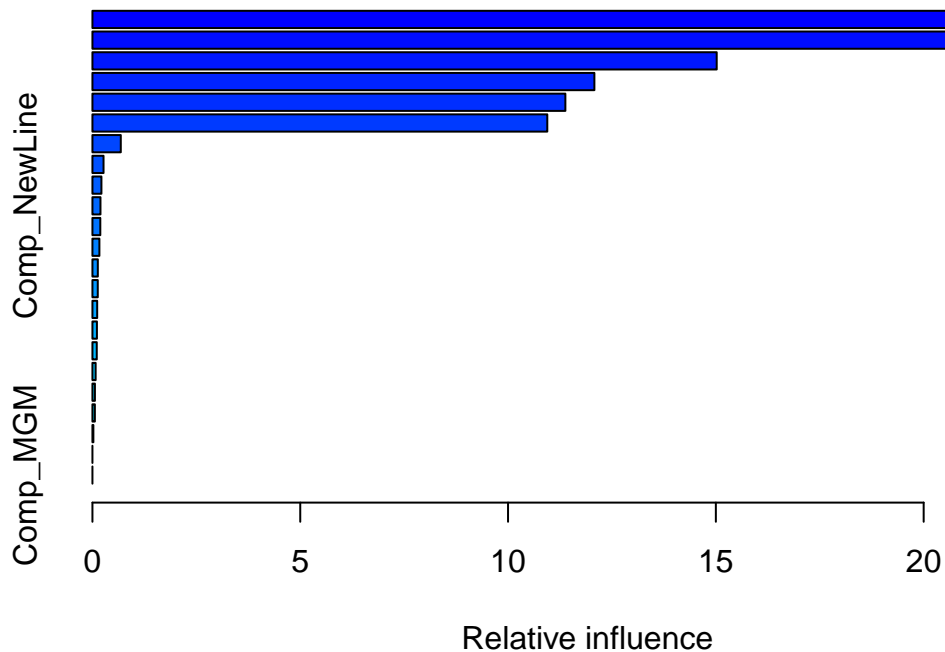
#Errors and best number of trees
err.2 = apply(yhat.boost.2, 2, function(pred) mean((train.2$vote_average - pred)^2))
best2=which.min(err.2)
min(err.2)

## [1] 0.4671261

best<- best2

#Final model with best number of trees on entire training set (dati.train)
boost.movies.2=gbm(vote_average ~ .-vote_classes, data=dati.train,
                    distribution="gaussian", n.trees=best2, interaction.depth=4)
summary(boost.movies.2)

```



```

##              var      rel.inf
## popularity    popularity 24.05904535
## genre         genre     24.00354584
## revenue       revenue   15.02038160
## runtime       runtime   12.07727602
## release_date  release_date 11.37929913
## budget       budget    10.94418395

```



```
## production_countries production_countries 0.68235369
## spoken_languages      spoken_languages 0.26674999
## Comp_20fox            Comp_20fox 0.21595410
## Comp_NewLine          Comp_NewLine 0.19291180
## Comp_DreamWorks       Comp_DreamWorks 0.18936393
## Comp_CanalPlus        Comp_CanalPlus 0.16719896
## Comp_Paramount        Comp_Paramount 0.12984632
## Comp_Universal        Comp_Universal 0.12834102
## Comp_Columbia         Comp_Columbia 0.11409711
## Comp_Touchstone       Comp_Touchstone 0.10770166
## Comp_Warner           Comp_Warner 0.10374298
## Comp_RelMedia         Comp_RelMedia 0.07699524
## Comp_Disney           Comp_Disney 0.05988659
## Comp_Miramax          Comp_Miramax 0.05856284
## Comp_VillageRoadshow  Comp_VillageRoadshow 0.02256188
## Comp_Pixar            Comp_Pixar 0.00000000
## Comp_MGM              Comp_MGM 0.00000000
```

```
#Prediction on test set
p.boost=predict(boost.movies.2, newdata=dati.test, n.trees=best)
dev.boost.2 <- sum((p.boost-dati.test$vote_average)^2)
#The previous model is better
dev.boost.2
```

```
## [1] 444.7384
```

LAB 1

```
library("readxl")
apple<- read_excel("DatiAPPLE.xlsx")
str(apple)
```

```
## tibble [56 x 4] (S3: tbl_df/tbl/data.frame)
## $ iPhone: num [1:56] 0.27 1.12 2.32 1.7 0.72 6.89 4.36 3.79 5.21 7.37 ...
## $ iPad : num [1:56] 3.27 4.19 7.33 4.69 9.25 ...
## $ iPod : num [1:56] 14.04 8.53 8.11 8.73 21.07 ...
## $ iMac : num [1:56] 1.25 1.11 1.33 1.61 1.61 ...
```

```
apple$iPhone
```

```
## [1] 0.27 1.12 2.32 1.70 0.72 6.89 4.36 3.79 5.21 7.37 8.74 8.75
## [13] 8.40 14.10 16.24 18.65 20.34 17.07 37.04 35.06 26.03 26.91 47.79 37.43
## [25] 31.24 33.80 51.03 43.72 35.20 39.27 74.47 61.17 47.53 48.05 74.78 51.19
## [37] 40.40 45.51 78.29 50.76 41.03 46.68 77.32 52.22 41.30 46.89 NA NA
## [49] NA NA NA NA NA NA NA NA
```

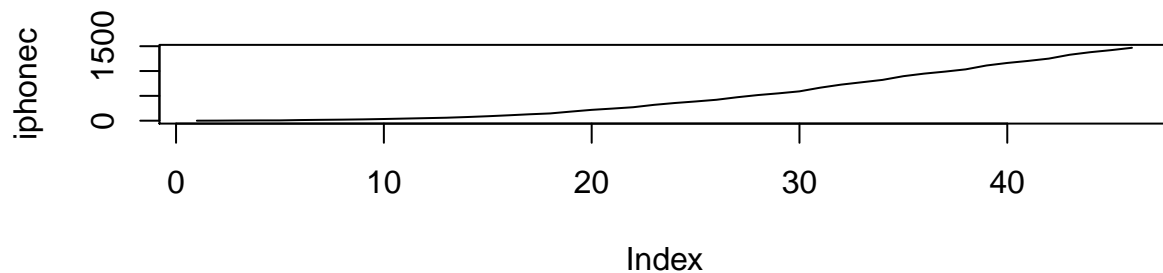
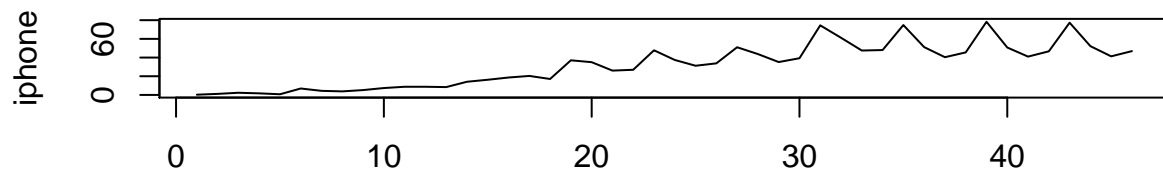
```
#Data cleaning (without Nan values) and cumulative data
iphone<- apple$iPhone[1:46]
iphone
```

```
## [1] 0.27 1.12 2.32 1.70 0.72 6.89 4.36 3.79 5.21 7.37 8.74 8.75
## [13] 8.40 14.10 16.24 18.65 20.34 17.07 37.04 35.06 26.03 26.91 47.79 37.43
## [25] 31.24 33.80 51.03 43.72 35.20 39.27 74.47 61.17 47.53 48.05 74.78 51.19
## [37] 40.40 45.51 78.29 50.76 41.03 46.68 77.32 52.22 41.30 46.89
```

```
iphonec<- cumsum(iphone)
iphonec
```

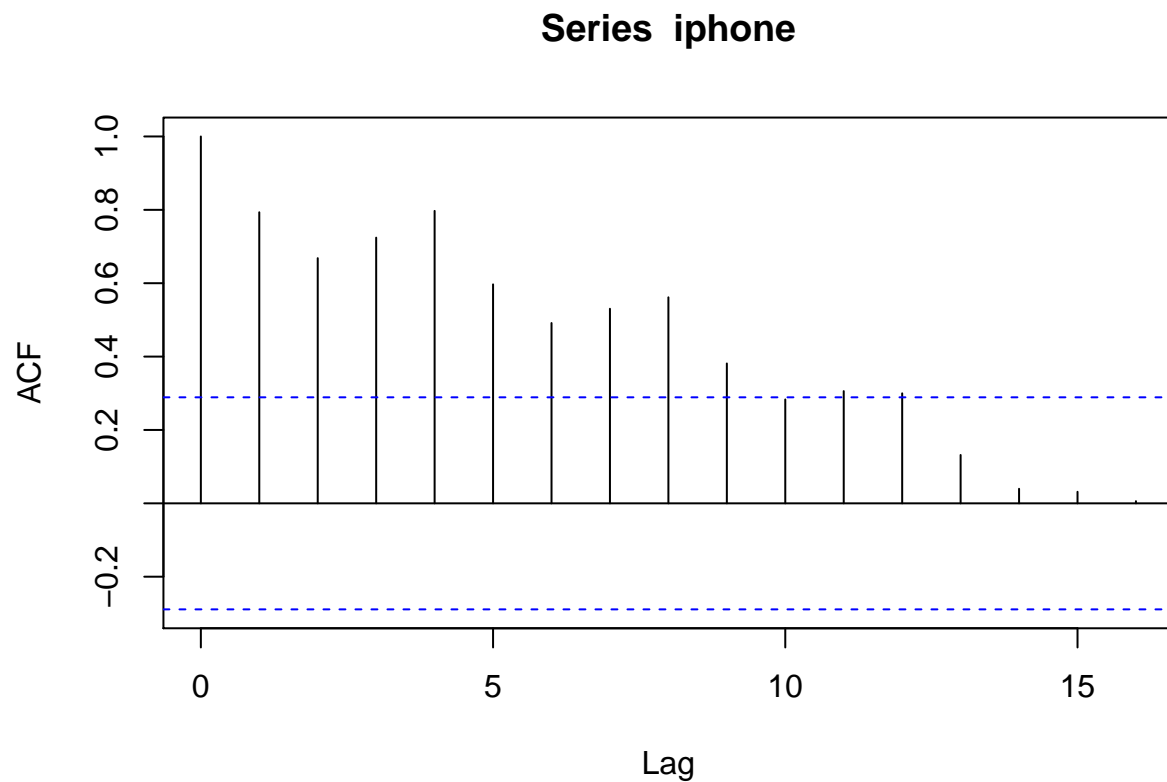
```
## [1] 0.27 1.39 3.71 5.41 6.13 13.02 17.38 21.17 26.38
## [10] 33.75 42.49 51.24 59.64 73.74 89.98 108.63 128.97 146.04
## [19] 183.08 218.14 244.17 271.08 318.87 356.30 387.54 421.34 472.37
## [28] 516.09 551.29 590.56 665.03 726.20 773.73 821.78 896.56 947.75
## [37] 988.15 1033.66 1111.95 1162.71 1203.74 1250.42 1327.74 1379.96 1421.26
## [46] 1468.15
```

```
#Plots
par(mfcol=c(2,1))
plot(iphone, type="l")
plot(iphonec, type="l")
```



```
par(mfrow=c(1,1))
```

```
#Autocorrelation function  
acf(iphone)
```



```
summary(iphone)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   
##  0.270   8.742   35.130   31.916   47.370   78.290
```

BASS model

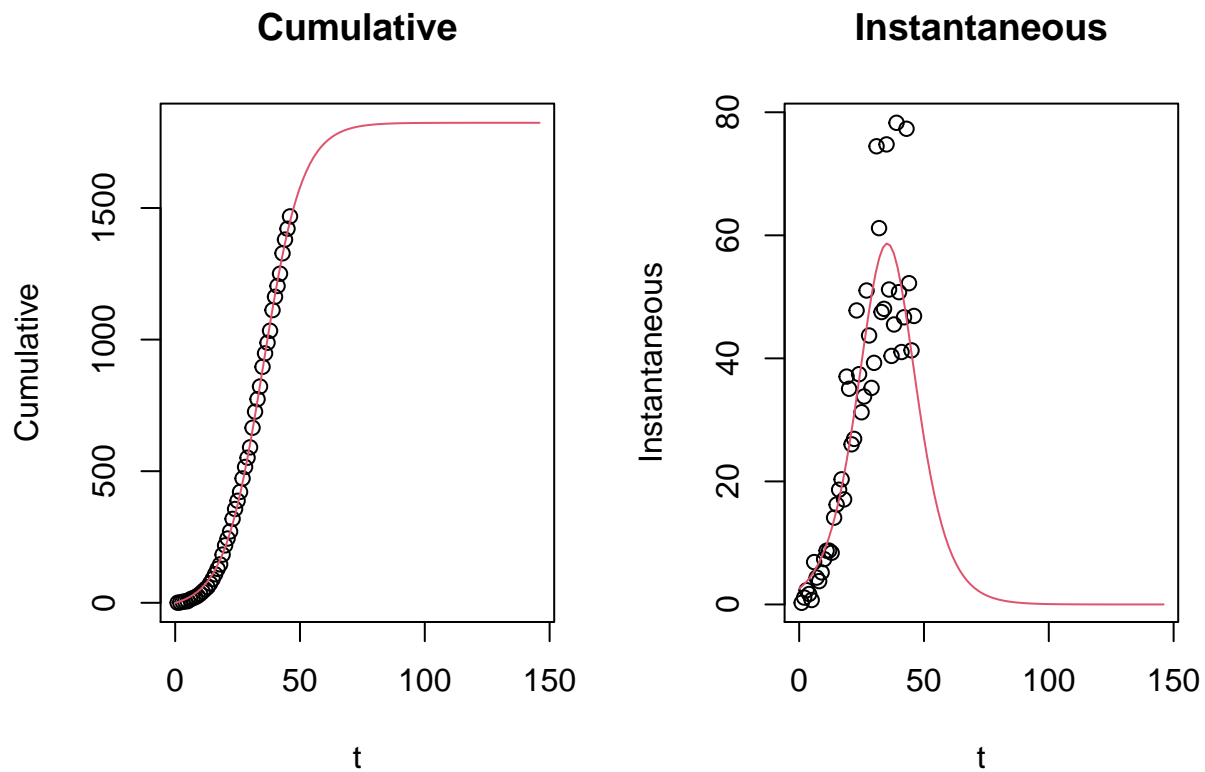
```
library(DIMORA)
```

```
## Loading required package: minpack.lm
```

```
## Loading required package: numDeriv
```

```
source("DIMORA1.0.0.R")
```

```
#Bass standard (forecasting)  
BMs<-BASS.standard(iphone,display = T)
```

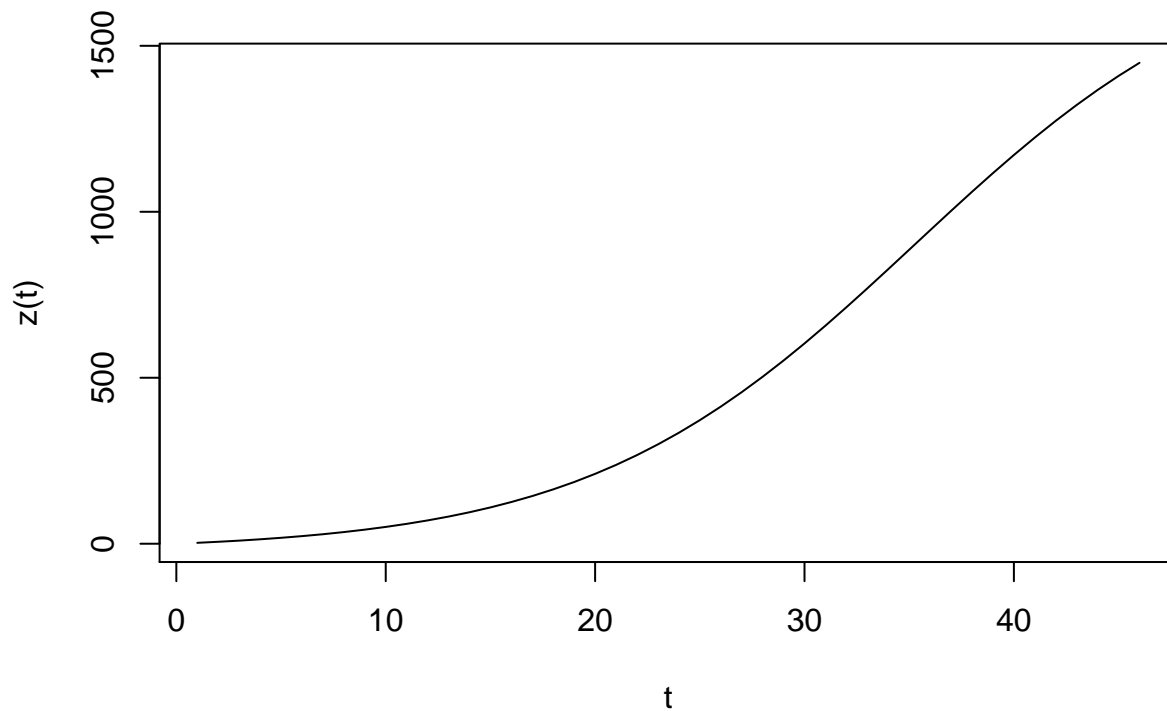


BMs

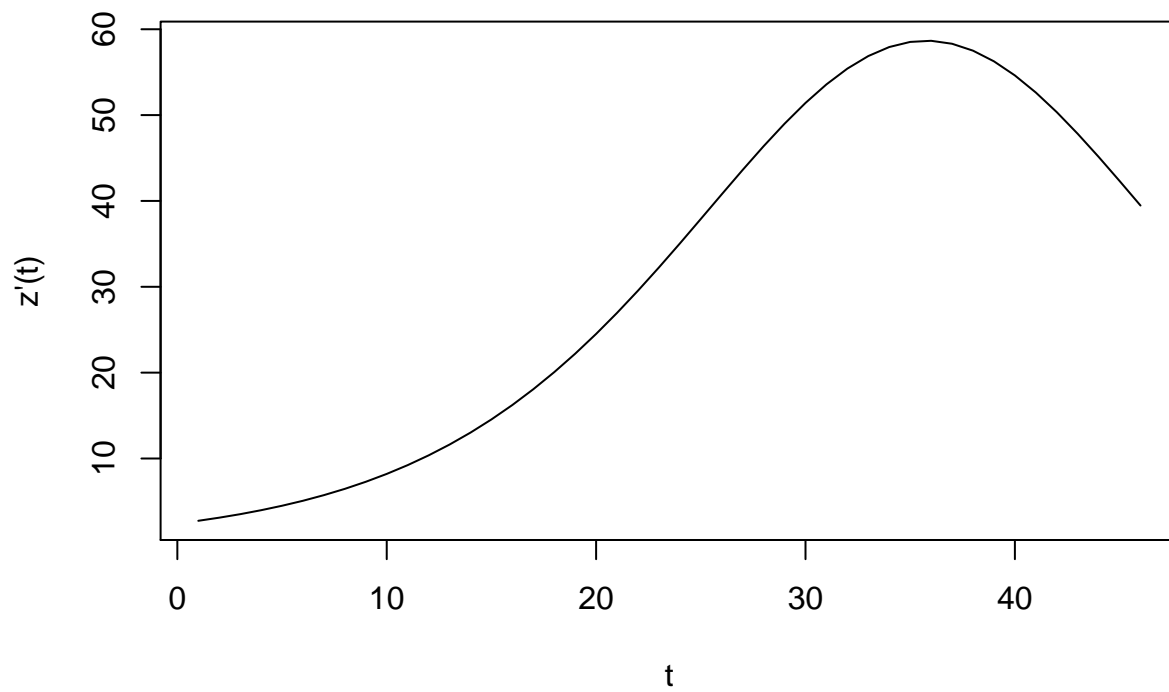
```
## $Estimate
##      Estimate      Std.Error      Lower      Upper p-value
## m : 1.823747e+03 3.412507e+01 1.756863e+03 1.890631e+03 5.84e-41
## p : 1.412817e-03 5.410927e-05 1.306765e-03 1.518869e-03 5.26e-28
## q : 1.258732e-01 2.675751e-03 1.206289e-01 1.311176e-01 1.29e-38
##
## $Rsquared
## [1] 0.9995498
##
## $RsquaredAdj
## [1] 0.9995176
##
## $residuals
## [1] -2.4736552 -4.4597443 -5.6546333 -7.9301912 -11.7043214 -9.8915082
## [7] -11.2633680 -13.9391872 -16.0164270 -16.8511654 -17.3384412 -18.9524544
## [13] -22.1765686 -21.0930498 -19.4024669 -16.9826675 -14.7072388 -17.6933582
## [19] -2.8589434 7.6909734 6.7587000 4.1191742 19.6560427 22.0384011
## [25] 15.3779786 8.4064478 15.8224349 13.1677182 -0.6279586 -12.7740436
## [31] 8.1211619 13.8779328 4.5294297 -5.3470600 10.9083851 3.4447115
## [37] -14.4649890 -26.4586382 -4.4295896 -8.2893669 -19.8879729 -23.5512761
## [43] 5.9447294 13.0327166 12.0055636 19.4301643
##
## $fitted
## [1] 2.743655 5.849744 9.364633 13.340191 17.834321 22.911508
```

```
## [7] 28.643368 35.109187 42.396427 50.601165 59.828441 70.192454
## [13] 81.816569 94.833050 109.382467 125.612667 143.677239 163.733358
## [19] 185.938943 210.449027 237.411300 266.960826 299.213957 334.261599
## [25] 372.162021 412.933552 456.547565 502.922282 551.917959 603.334044
## [31] 656.908838 712.322067 769.200570 827.127060 885.651615 944.305289
## [37] 1002.614989 1060.118638 1116.379590 1170.999367 1223.627973 1273.971276
## [43] 1321.795271 1366.927283 1409.254436 1448.719836
```

```
#The function "fitted()" gives us a cumulative output by default, to use the "normal" fit we should use
"make.instantaneous()" function
pred_BM<- fitted(BMs)
#Cumulative forecasting
plot(pred_BM, type="l",ylab="z(t)",xlab="t")
```

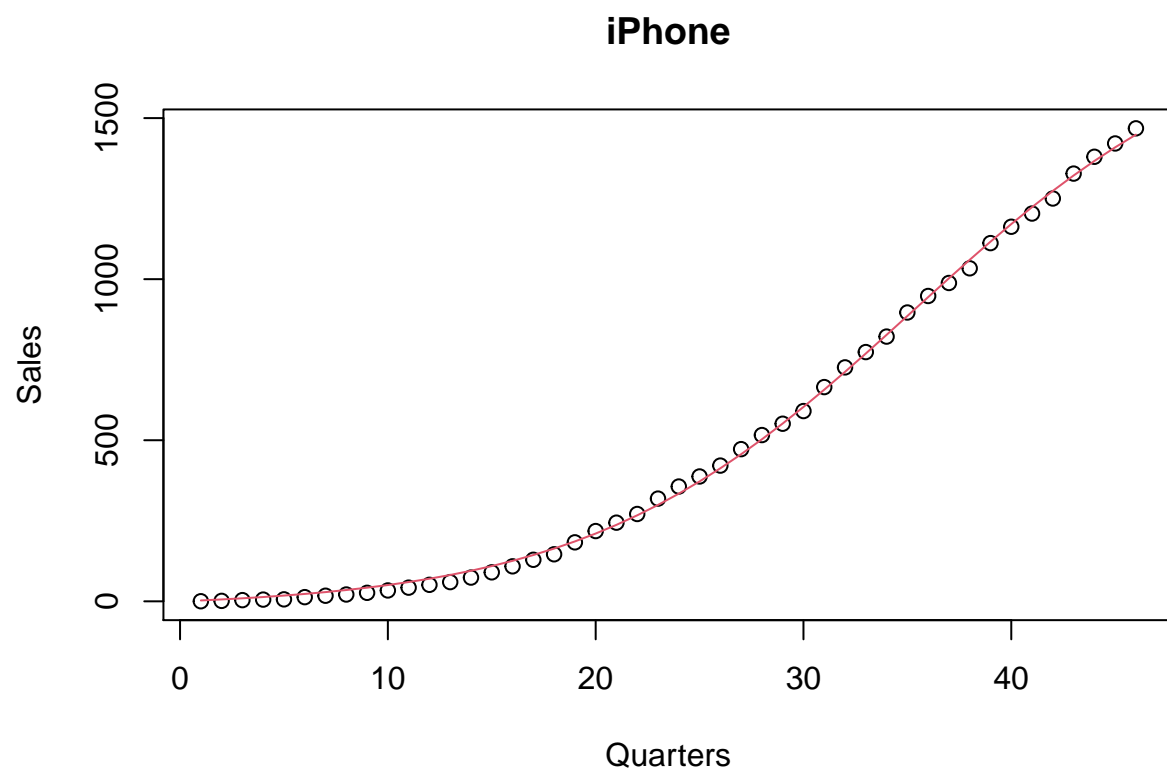


```
#Instantaneous forecasting
pred_BMinst<- make.instantaneous(pred_BM)
plot(pred_BMinst, type="l",ylab="z'(t)",xlab="t")
```

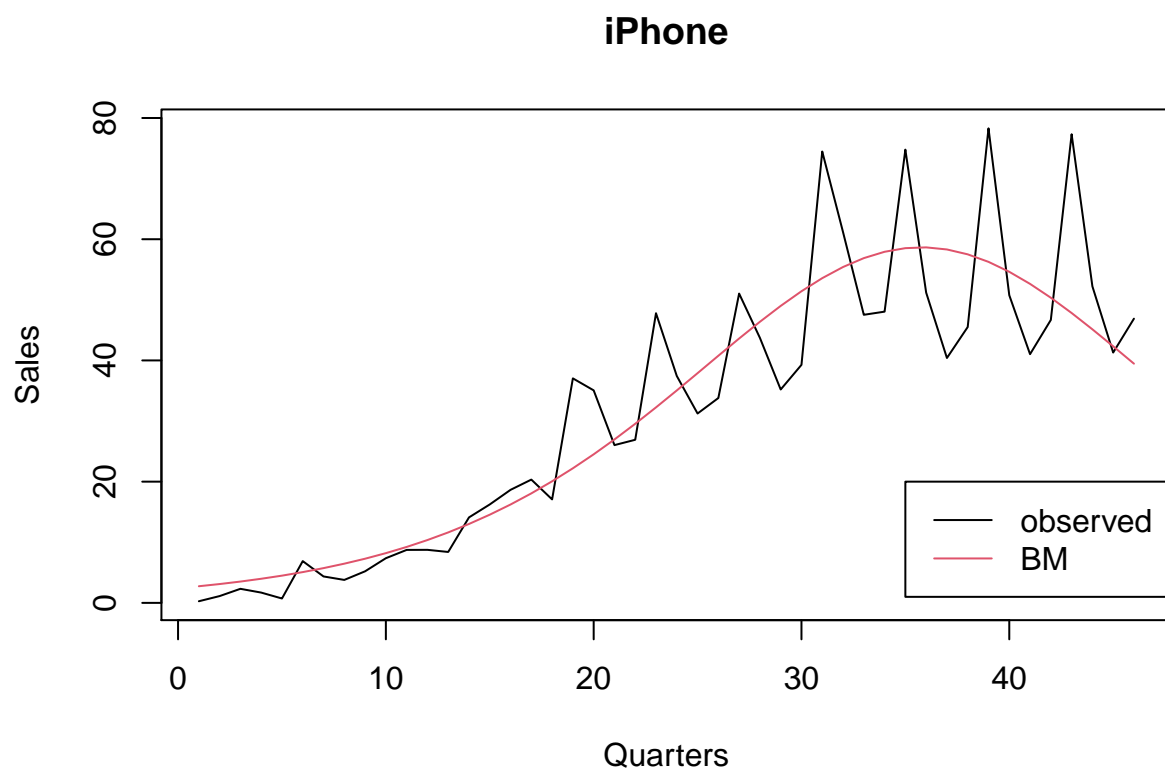


```
#Change the character of labels
#plot(pred_BMinst, type="l", ylab=expression(italic("z'(t)")), xlab=expression(italic(t)))
```

```
#Cumulative data plot
plot(iphonec, xlab="Quarters", ylab="Sales", main="iPhone")
lines(pred_BM, col=2)
```



```
#Instantaneous data plot  
plot(iphone, type= "l",xlab="Quarters", ylab="Sales", main="iPhone")  
lines(pred_BMinst,col=2)  
legend(35,20,legend= c("observed","BM"), col=c(1,2), lty=1)
```



```
#Bass Model estimates
```

```
BM$s$Estimate
```

```
##      Estimate  Std.Error      Lower      Upper  p-value
## m : 1.823747e+03 3.412507e+01 1.756863e+03 1.890631e+03 5.84e-41
## p : 1.412817e-03 5.410927e-05 1.306765e-03 1.518869e-03 5.26e-28
## q : 1.258732e-01 2.675751e-03 1.206289e-01 1.311176e-01 1.29e-38
```

```
m<-BM$s$Estimate[1,1]
```

```
m
```

```
## [1] 1823.747
```

```
p<-BM$s$Estimate[2,1]
```

```
p
```

```
## [1] 0.001412817
```

```
q<-BM$s$Estimate[3,1]
```

```
q
```

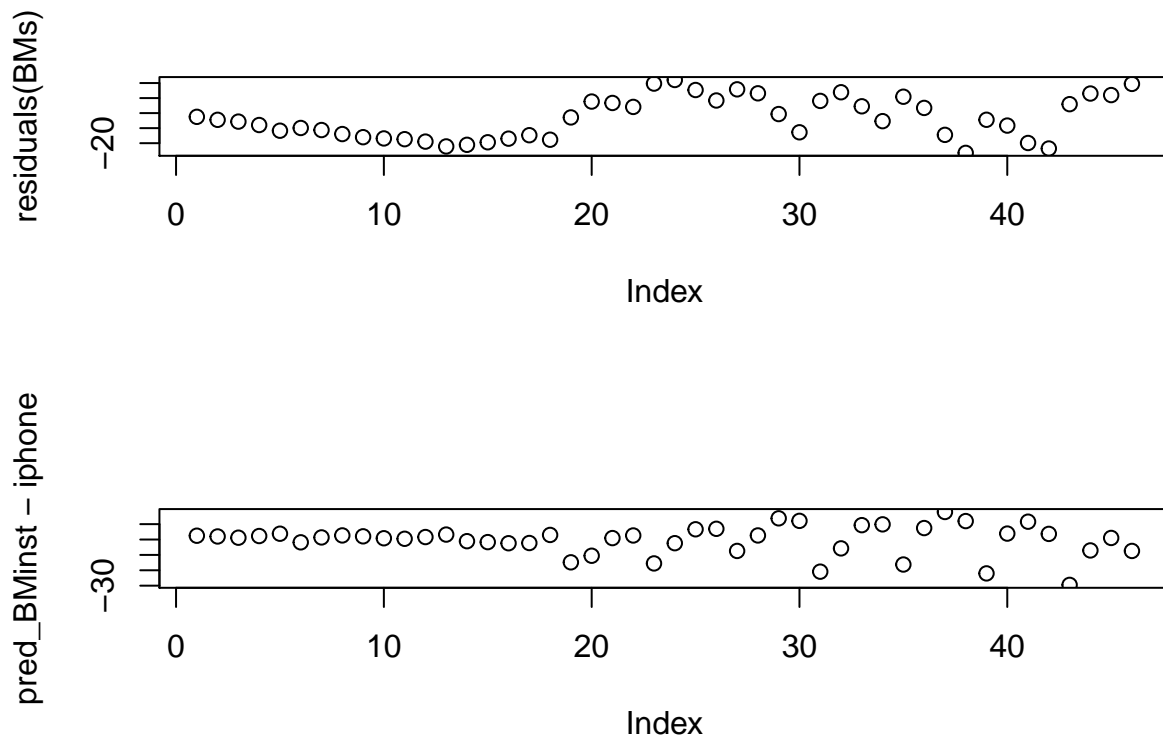
```
## [1] 0.1258732
```



```

#Residuals
par(mfrow=c(2,1))
#Cumulative (the residuals are not random, there is an oscillatory pattern)
plot(residuals(BMs))
#Instantaneous
plot(pred_BMinst-iphone)

```



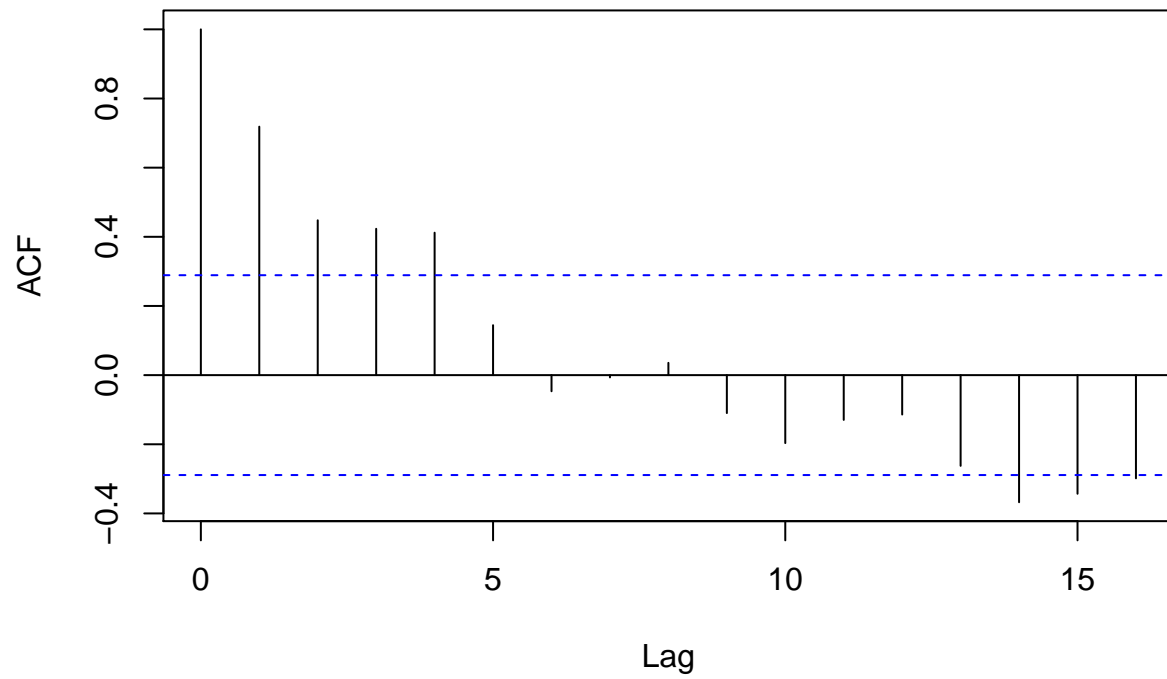
Autocorrelation function (this function confirms the correlation between residuals) first five residuals are significant correlated (also the 14th and 15th)

```

#Cumulative data
acf(residuals(BMs))

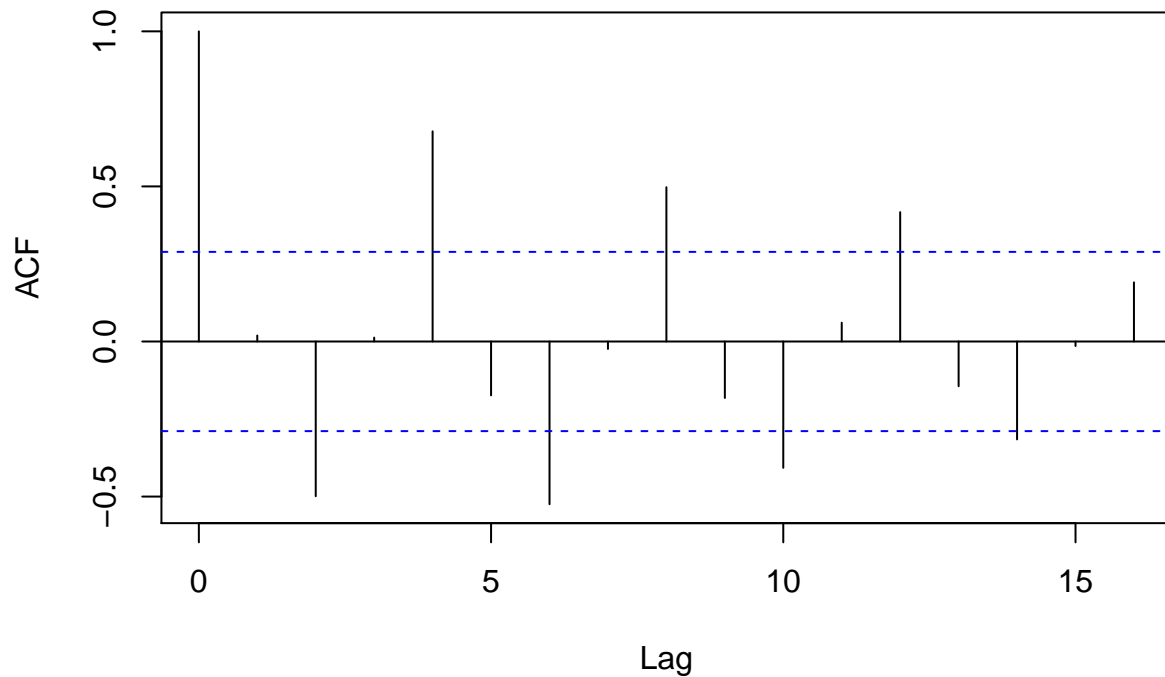
```

Series residuals(BMs)



```
#Instantaneous data (another confirmation of relationship)  
acf(pred_BMinst-iphone)
```

Series pred_BMinst – iphone



```
par(mfrow=c(1,1))
```

GENERALIZED BASS MODEL

Generalized Bass Model (with only one exponential shock) m , p and q derive from the previous Bass Model, we use them like a starting point; 17 (a1): 17th observation, starting point (moment) of the shock (in the slides the start. point is different -> so the estimation is different, a good choice of $a1$ depends from our observation of sales distribution, we hypothesize a shock in a particular moment seeing the plot) -0.1 (b1): memory of the shock, in this case it is a negative memory (it is a careful starting point) 0.1 (c1): intensity of the shock (it is a careful starting point)

```
GBMe1<-BASS.generalized(iphone, shock = "exp", nshock = 1, prelinestimates = c(m, p , q, 17, -0.1, 0.1))
```

```
## ##### Exponential shock #####
```

```
GBMe1
```

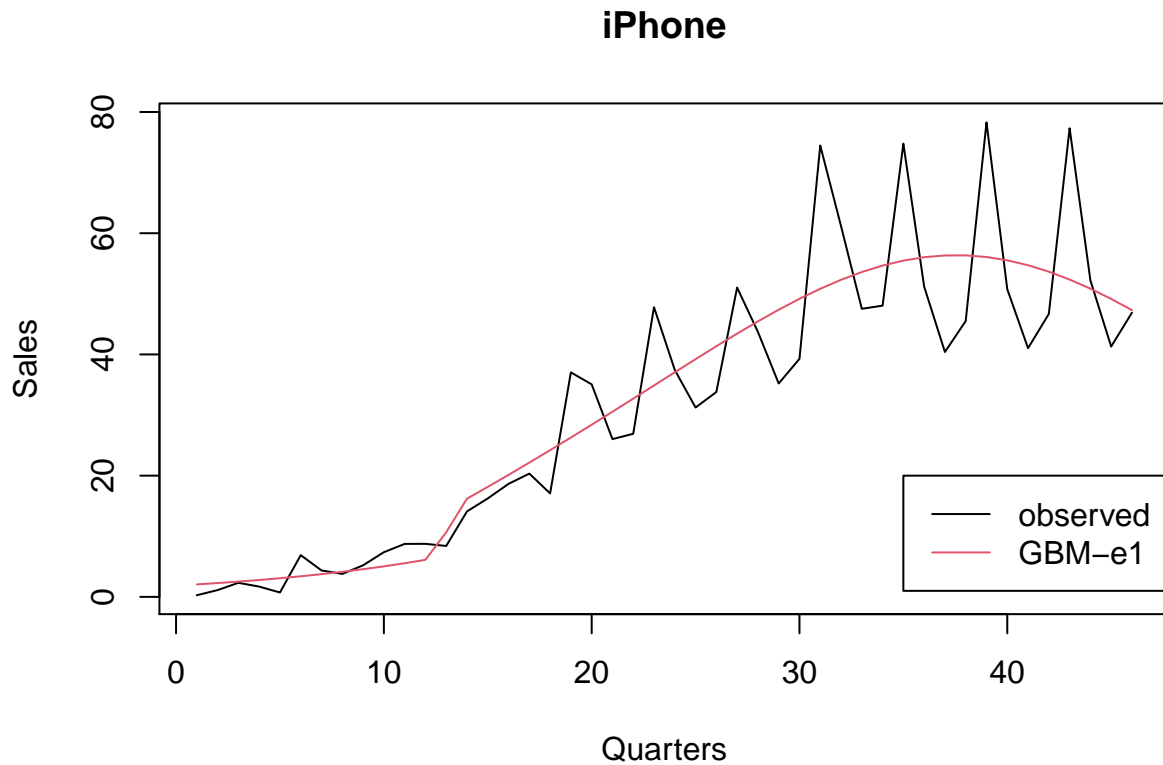
```
## $Estimate
```

##	Estimate	Std.Error	Lower	Upper	P-value
## m :	2.108930e+03	1.249330e+02	1.864066e+03	2353.79429206	8.51e-20
## p :	9.298680e-04	9.280907e-05	7.479656e-04	0.00111177	1.83e-12
## q :	1.014024e-01	1.119414e-02	7.946233e-02	0.12334254	3.09e-11
## a1 :	1.250223e+01	9.891378e-01	1.056356e+01	14.44090701	1.50e-15

```
## b1 : -1.382643e-01 5.742298e-02 -2.508113e-01 -0.02571732 2.08e-02
## c1 : 1.127078e+00 1.748176e-01 7.844417e-01 1.46971418 1.11e-07
##
## $Rsquared
## [1] 0.9998677
##
## $RsquaredAdj
## [1] 0.9998473
##
## $RSS
## [1] 2649.878
##
## $residuals
## [1] -1.79285721 -2.95328425 -3.15367456 -4.23860267 -6.59500887
## [6] -3.10239321 -2.49301758 -2.84211431 -2.19809896 0.13721566
## [11] 3.32840839 5.96624630 3.71156672 1.60904877 -0.29490170
## [16] -1.77165483 -3.57848492 -10.70989653 0.04193538 6.69756918
## [21] 2.18145652 -3.61625802 9.29257005 9.66675636 1.68867308
## [26] -5.86280771 1.73052596 -0.00120183 -12.17381397 -22.07752788
## [31] 1.56386627 10.42297879 4.35786157 -2.24992174 17.05181598
## [36] 12.20210788 -3.72770856 -14.55912176 7.65805173 2.89003501
## [41] -10.79642030 -17.76918323 7.19468368 8.56510005 0.70601343
## [46] 0.28337968
##
## $fitted
## [1] 2.062857 4.343284 6.863675 9.648603 12.725009 16.122393
## [7] 19.873018 24.012114 28.578099 33.612784 39.161592 45.273754
## [13] 55.928433 72.130951 90.274902 110.401655 132.548485 156.749897
## [19] 183.038065 211.442431 241.988543 274.696258 309.577430 346.633244
## [25] 385.851327 427.202808 470.639474 516.091202 563.463814 612.637528
## [31] 663.466134 715.777021 769.372138 824.029922 879.508184 935.547892
## [37] 991.877709 1048.219122 1104.291948 1159.819965 1214.536420 1268.189183
## [43] 1320.545316 1371.394900 1420.553987 1467.866620
```

```
pred_GBMe1<- fitted(GBMe1)
pred_GBMinst<-make.instantaneous(pred_GBMe1)
```

```
#Plot
plot(iphone, type= "l",xlab="Quarters", ylab="Sales", main="iPhone")
lines(pred_GBMinst,col=2)
legend(35,20,legend= c("observed","GBM-e1"), col=c(1,2), lty=1)
```



if $R2_tilde > 0.3$, the more complex model is significant (in this case GBM is better than BM) it makes sense to use a more complex model

```
R2_tilde <- (GBMe1$Rsquared - BMs$Rsquared) / (1 - BMs$Rsquared)
R2_tilde
```

```
## [1] 0.7061501
```

GUISEO-GUIDOLIN MODEL

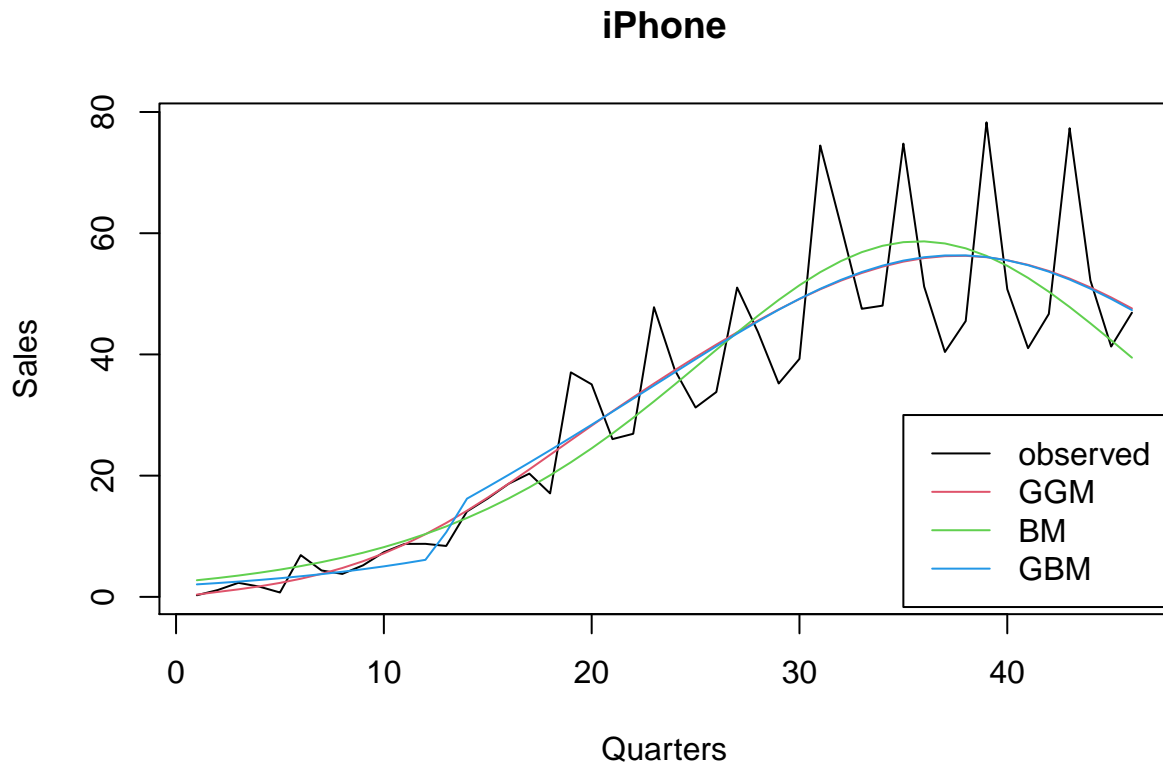
```
GGM <- GG.model(iphone, prlimestimates = c(m, 0.01, 0.1, p, q), display = F)
GGM
```

```
## $Estimate
##           Estimate   Std.Error      Lower      Upper P-value
## k : 2.116781e+03 9.750237e+01 1.925679e+03 2.307882e+03 4.23e-24
## pc : 5.923751e-03 1.592869e-03 2.801784e-03 9.045717e-03 6.00e-04
## qc : 2.055800e-01 3.778092e-02 1.315308e-01 2.796293e-01 2.69e-06
## ps : 2.124610e-03 2.768449e-04 1.582004e-03 2.667216e-03 1.87e-09
## qs : 1.001408e-01 7.445880e-03 8.554716e-02 1.147345e-01 1.26e-16
##
## $Rsquared
## [1] 0.9998694
```

```
##
## $RsquaredAdj
## [1] 0.9998531
##
## $residuals
## [1] -0.1124894  0.1924584  1.2709098  1.2412462 -0.3442201  3.5559637
## [7]  4.1137696  3.1426756  2.4693853  2.6568906  2.7292407  1.1381148
## [13] -2.6590625 -2.7801540 -2.9295897 -2.9512270 -3.6440407 -10.0114335
## [19]  1.1757981  7.9836281  3.3974810 -2.6249803  9.9703939 10.0007859
## [25]  1.6954804 -6.1328186  1.2549820 -0.6017261 -12.8190908 -22.6949624
## [31]  1.0341808 10.0252449  4.1192950 -2.3189802 17.1469050 12.4422079
## [37] -3.3731042 -14.1292389  8.1179705  3.3313280 -10.4234585 -17.5132325
## [43]  7.2876820  8.4532775  0.3525322 -0.3429638
##
## $fitted
## [1]  0.3824894  1.1975416  2.4390902  4.1687538  6.4742201
## [6]  9.4640363 13.2662304 18.0273244 23.9106147 31.0931094
## [11] 39.7607593 50.1018852 62.2990625 76.5201540 92.9095897
## [16] 111.5812270 132.6140407 156.0514335 181.9042019 210.1563719
## [21] 240.7725190 273.7049803 308.8996061 346.2992141 385.8445196
## [26] 427.4728186 471.1150180 516.6917261 564.1090908 613.2549624
## [31] 663.9958192 716.1747551 769.6107050 824.0989802 879.4130950
## [36] 935.3077921 991.5231042 1047.7892389 1103.8320295 1159.3786720
## [41] 1214.1634585 1267.9332325 1320.4523180 1371.5067225 1420.9074678
## [46] 1468.4929638
```

```
pred_GGM <- fitted(GGM)
pred_GGMinst <- make.instantaneous(pred_GGM)
```

```
#Plot comparison between GGM, BM and GBM
plot(iphone, type= "l",xlab="Quarters", ylab="Sales", main="iPhone")
lines(pred_GGMinst,col=2)
lines(pred_BMinst,col=3)
lines(pred_GBMinst, col=4)
legend(35,30,legend= c("observed","GGM", "BM", "GBM"), col=c(1,2,3,4), lty=1)
```



ARIMA MODEL

Considering the fact that our GGM is good but doesn't capture the high variability (see graph) of the data (the peaks are ignored by our GGM). We will try to model the residuals with the ARIMA model.

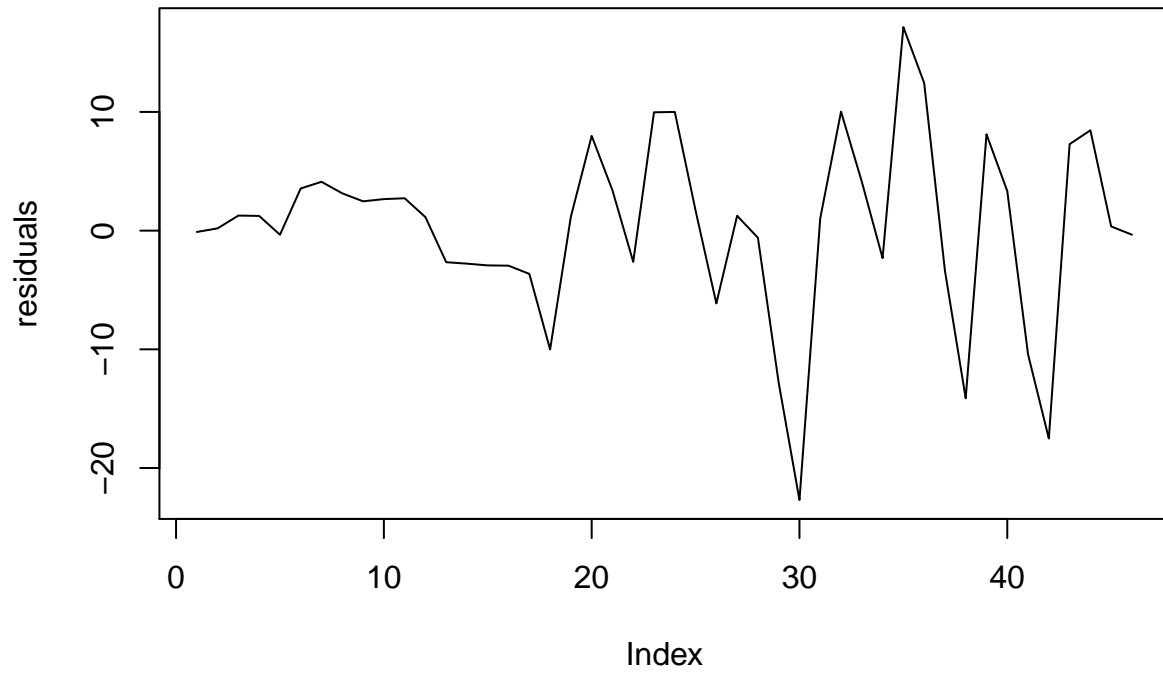
```
library(forecast)
#We will use the residuals obtained from GGM
residuals <- residuals(GGM)
residuals
```

```
## [1] -0.1124894  0.1924584  1.2709098  1.2412462 -0.3442201  3.5559637
## [7]  4.1137696  3.1426756  2.4693853  2.6568906  2.7292407  1.1381148
## [13] -2.6590625 -2.7801540 -2.9295897 -2.9512270 -3.6440407 -10.0114335
## [19]  1.1757981  7.9836281  3.3974810 -2.6249803  9.9703939  10.0007859
## [25]  1.6954804 -6.1328186  1.2549820 -0.6017261 -12.8190908 -22.6949624
## [31]  1.0341808 10.0252449  4.1192950 -2.3189802 17.1469050 12.4422079
## [37] -3.3731042 -14.1292389  8.1179705  3.3313280 -10.4234585 -17.5132325
## [43]  7.2876820  8.4532775  0.3525322 -0.3429638
```

The residuals don't show any trend. This is due to the fact that the GGM had captured the trend in time series. We have already modeled our trend, so in the residuals there isn't any trend. This is a good result, we expected it using GGM.

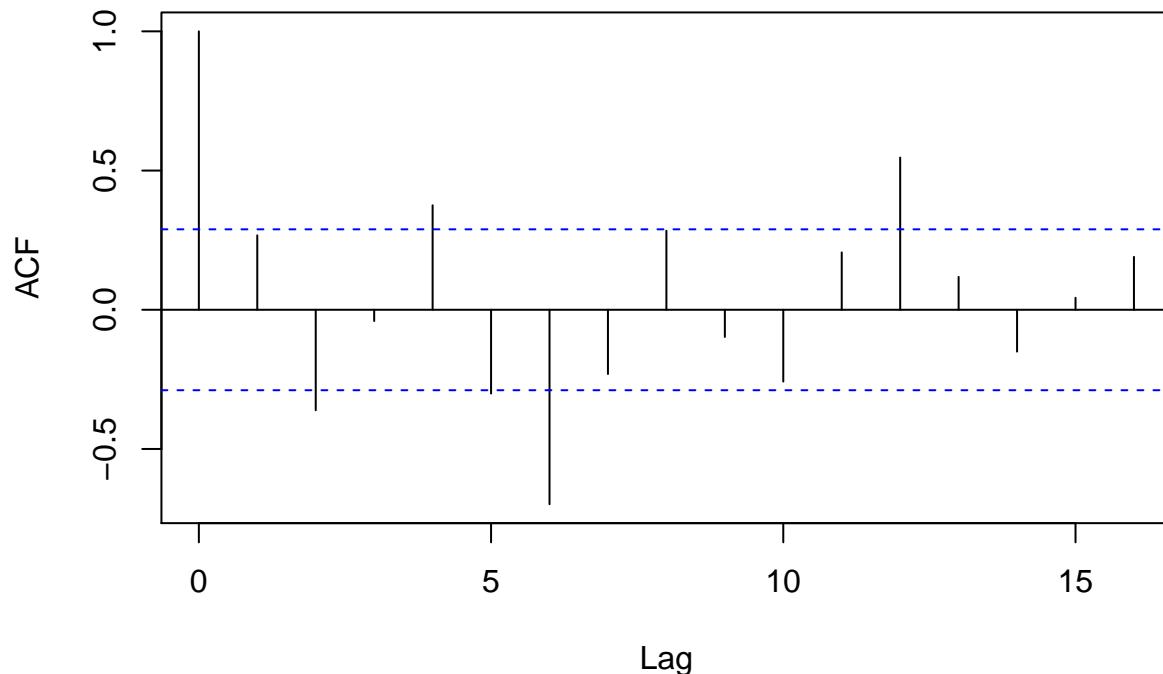
A part from few lag residuals, the graph doesn't show any particular trend in autocorr., it is coherent with previous result. But it's important to notice that there are significant autocorrelations that show a behavior that we have to model.

```
plot(residuals, type= 'l')
```



```
acf(residuals)
```


Series residuals



ARIMA model on residuals with ‘manual’ selection We try to model the residuals (our new time series) (instead of the data itself). Arima’s arguments: time series, order, seasonal and period. In “order” we consider: the two ones refer to the autoregressive and moving average element, the zero (differencing) means that we deal with a t.s. without any trend, so the diff. isn’t useful (remember that the residuals don’t show any particular trend). In any case, we consider the seasonality (?) part because our series is characterized by seasonality in any case. In this case “period” refers to quarterly data (three months).

```
arimaResiduals <- Arima(residuals, order = c(1,0,1), seasonal = list(order = c(1,0,0), period = 4))
```

```
#ARIMA model for residuals with auto.arima
#autoarima<- auto.arima(residuals)
#Best model selected with AIC (not always)
#autoarima
```

```
#ar1: coeff. autoregressive non-seasonal part
#ma1: coeff. moving average non-seasonal part
#sar1: coeff. aut. seasonal part
#mean: mean behavior
#To see the significance of our coefficients, we compute: coeff./s.e.. If the division is
#greater than two, it's significant (in this case all, except the mean, are sign.).
summary(arimaResiduals)
```

```
## Series: residuals
## ARIMA(1,0,1)(1,0,0)[4] with non-zero mean
##
## Coefficients:
```

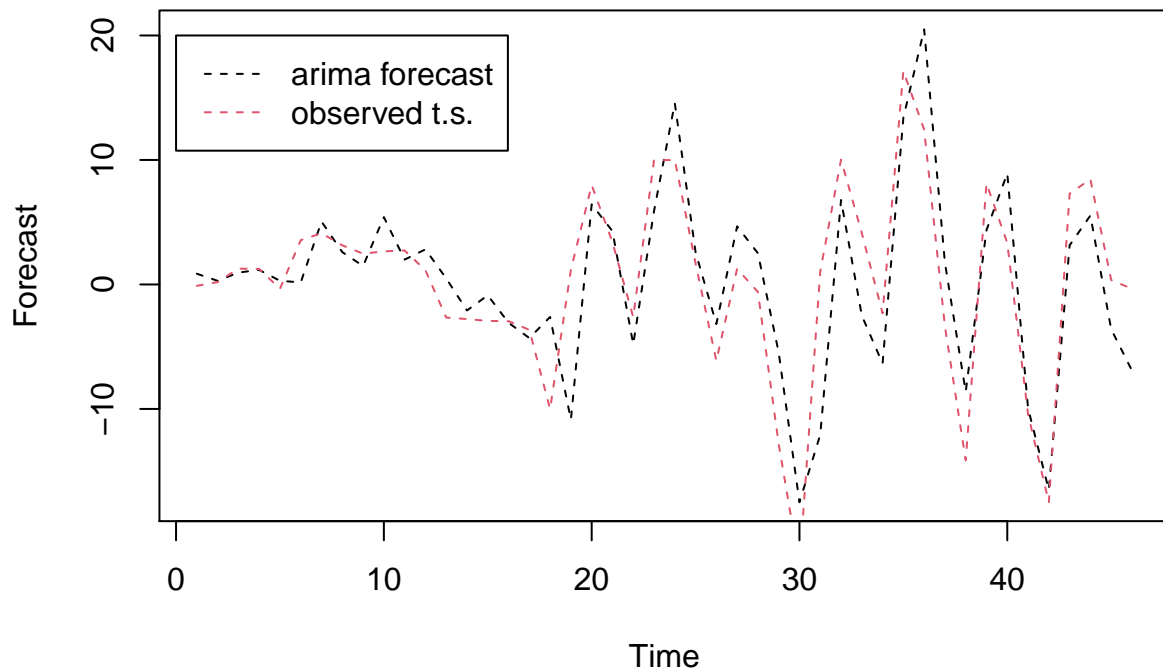
```
##          ar1      ma1      sar1      mean
##          0.6014  0.5406  0.8428   3.5094
## s.e.    0.1317  0.1230  0.0789  11.4540
##
## sigma^2 estimated as 21.6:  log likelihood=-137.05
## AIC=284.09   AICc=285.59   BIC=293.24
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 0.09306844 4.440626 3.331901 66.6587 203.5847 0.51836 0.08967188
```

```
#Forecasting on residuals and plot comparison
```

```
Forecast <- fitted(arimaResiduals)
Forecast
```

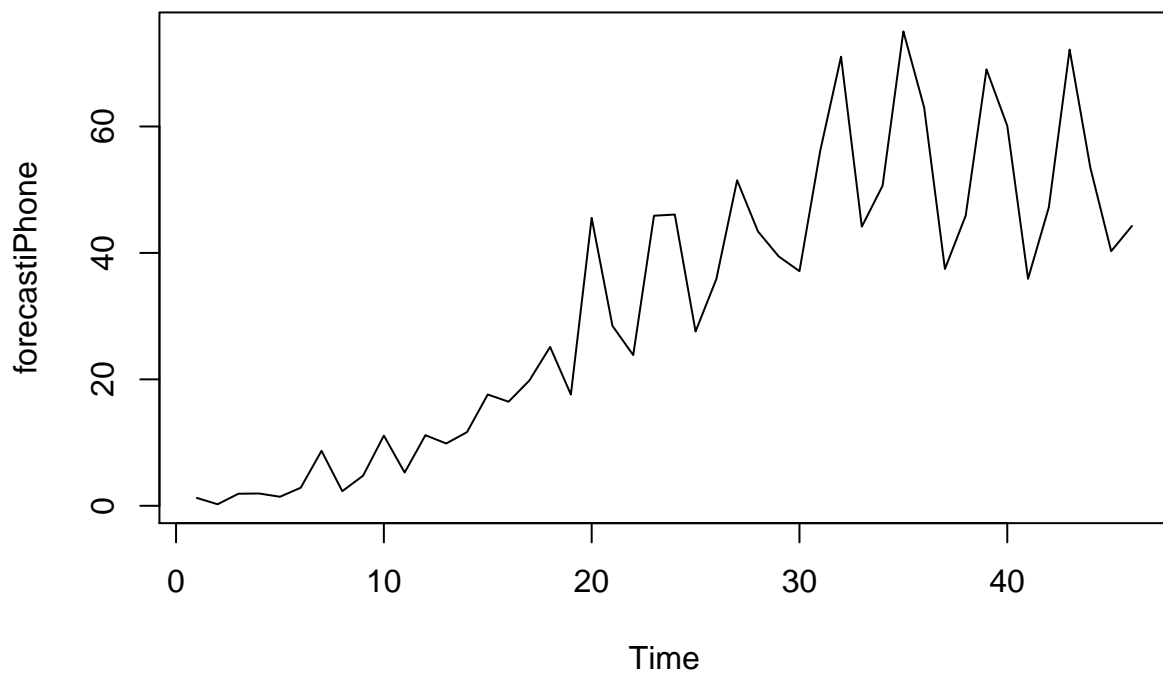
```
## Time Series:
## Start = 1
## End = 46
## Frequency = 1
## [1] 0.8599222 0.2912046 0.9549652 1.1693423 0.2927843 0.1645767
## [7] 5.0502045 2.5996697 1.4842151 5.4071812 1.9962279 2.8210644
## [13] 0.4829092 -2.0901001 -0.8714143 -3.0785452 -4.3037634 -2.6101586
## [19] -10.8618610 6.4324919 4.2842572 -4.8068203 5.8858046 14.5565106
## [25] 2.5882240 -3.1772212 4.6671423 2.5051942 -5.4612126 -17.4947232
## [31] -12.0735476 6.7851326 -2.4980942 -6.3550500 13.3810969 20.4922614
## [37] 1.7408831 -8.6153534 4.3683546 8.9246267 -9.9492133 -16.5029074
## [43] 3.1443638 5.5357679 -3.5921840 -6.9123027
```

```
plot(Forecast, lty=2)
lines(residuals, lty=2,col=2)
legend(0,20,legend= c("arima forecast", "observed t.s."), col=c(1,2), lty=2)
```

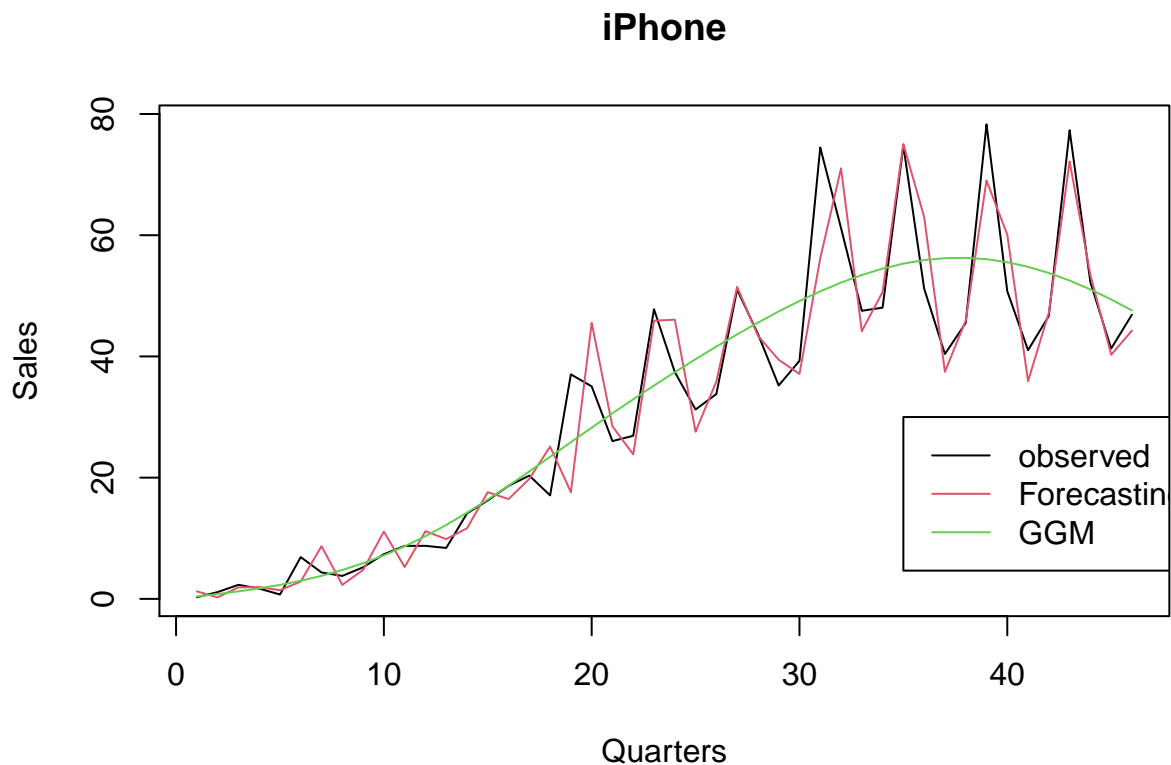


We use GGM and ARIMA to obtain a forecasting that exploits GGM forecasting on data and ARIMA forecasting on residuals

```
forecast.inst <- make.instantaneous(Forecast)
#Here we sum the predicted values using GGM and the forecasting on residuals obtained with ARIMA
forecastiPhone <- pred_GGMinst + forecast.inst
plot(forecastiPhone)
```



```
plot(iphone, type='l', xlab = "Quarters", ylab = "Sales", main = "iPhone")
lines(forecastiPhone, col=2)
lines(pred_GGMinst,col=3)
legend(35,30,legend= c("observed","Forecasting","GGM"), col=c(1,2,3), lty=1)
```



EXPLORATIVE PART (To see in practice other models, not necessary)

```
#Now we calculate a manual direct ARIMA and an auto direct ARIMA:
#Manual direct ARIMA (2 versions)
#Without differencing
directarima<- Arima(iphone, order = c(1,0,1), seasonal = list(order = c(1,0,0), period = 4))
summary(directarima)

## Series: iphone
## ARIMA(1,0,1)(1,0,0)[4] with non-zero mean
##
## Coefficients:
##          ar1          ma1          sar1          mean
##          0.8003      -0.1587      0.9185      28.6193
## s.e.    0.1165      0.2402      0.0463      23.0608
##
## sigma^2 estimated as 31.33: log likelihood=-146.97
## AIC=303.95   AICc=305.45   BIC=313.09
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 1.040555 5.348182 3.476329 -41.62963 59.66054 0.3731391
##              ACF1
## Training set -0.02879817
```

```
NewForecast <- fitted(directarima)
NewForecast
```

```
## Time Series:
## Start = 1
## End = 46
## Frequency = 1
## [1] 5.5558158 2.1030198 2.5974666 2.1771941 0.8285815 1.9132832
## [7] 6.5019849 4.1514298 2.9682962 10.0787926 5.7342634 7.2598367
## [13] 9.2316270 10.2601145 13.7510321 14.6803499 16.0453325 22.8387848
## [19] 19.5946720 32.5329403 33.0970651 23.1472930 42.8776561 42.9083132
## [25] 29.4279877 30.7625672 51.1473456 40.5744722 36.1361942 36.8665668
## [31] 53.5369659 59.3884342 49.3322048 48.9853498 78.6016623 62.3628574
## [37] 41.8984880 42.2310977 69.7304892 53.8126198 41.0528578 45.4093380
## [43] 76.0765692 51.2225890 42.4731611 46.4195228
```

```
#With differencing (d=1)
directarima_diff<- Arima(iphone, order = c(1,1,1), seasonal = list(order = c(1,0,0), period = 4))
summary(directarima_diff)
```

```
## Series: iphone
## ARIMA(1,1,1)(1,0,0)[4]
##
## Coefficients:
##          ar1          ma1          sar1
##          0.5632      -0.9190      0.9184
## s.e.    0.1835      0.0838      0.0519
##
## sigma^2 estimated as 30.19: log likelihood=-142.1
## AIC=292.21 AICc=293.21 BIC=299.44
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.6079901 5.250439 3.310344 5.727905 15.80766 0.3553226
##              ACF1
## Training set -0.006588959
```

```
NewForecast_diff <- fitted(directarima_diff)
NewForecast_diff
```

```
## Time Series:
## Start = 1
## End = 46
## Frequency = 1
## [1] 0.2697300 0.7985603 1.7286984 1.5669600 0.6122964 1.5159181
## [7] 6.1505555 3.3715316 2.5368223 9.7312925 5.2479982 7.1191614
## [13] 8.8622558 9.8920736 13.6085939 14.3480099 15.7493594 22.4880427
## [19] 19.2451139 33.0699188 32.4165894 22.9336234 43.7902624 43.7245027
## [25] 30.1076852 32.1935032 52.4866221 41.7579941 37.4735345 38.0435730
## [31] 54.9396409 60.7238029 49.2256833 49.5509095 79.9442637 64.1568899
## [37] 44.1720277 45.3220611 72.4954347 55.9350364 42.3031494 46.9941701
## [43] 77.6129730 52.6066012 43.7426017 47.6164280
```

```
#Auto direct ARIMA
```

```
autodirectarima<- auto.arima(iphone)
summary(autodirectarima)
```

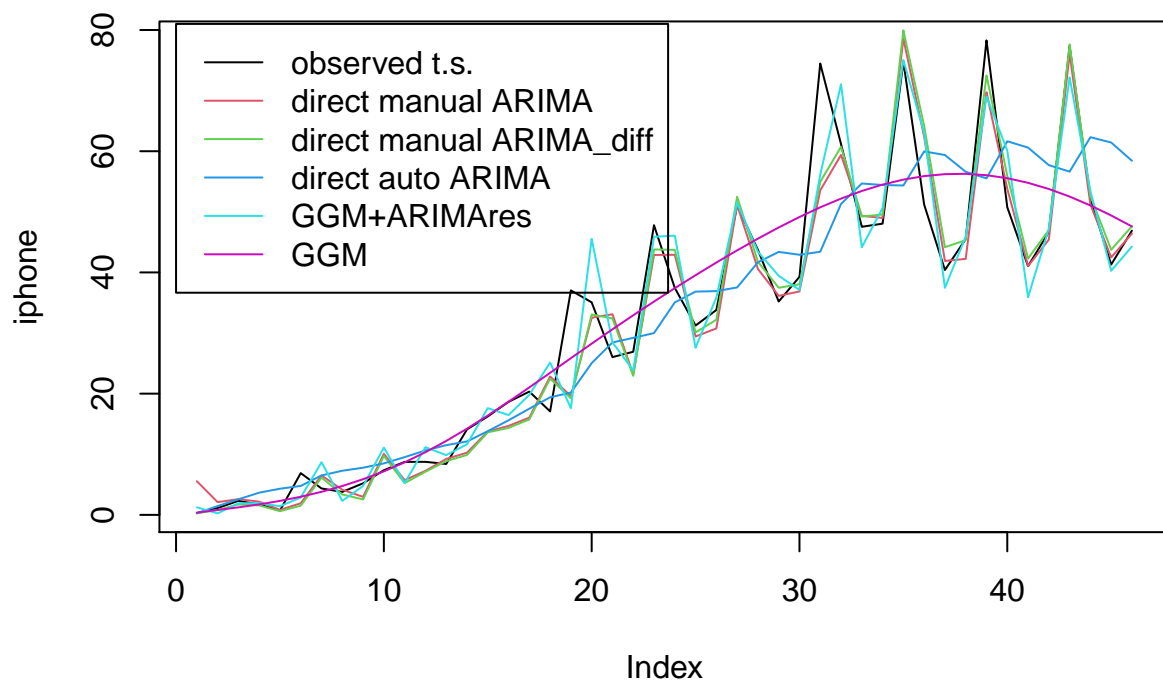
```
## Series: iphone
## ARIMA(0,1,1) with drift
##
## Coefficients:
##          ma1    drift
##       -0.7872  1.2737
## s.e.    0.0953  0.3941
##
## sigma^2 estimated as 127:  log likelihood=-172.3
## AIC=350.6   AICc=351.19   BIC=356.02
##
## Training set error measures:
##              ME    RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set -0.09961446 10.8942  7.918186 -20.4835 36.99582  0.849915  0.05672631
```

```
NewForecast2 <- fitted(autodirectarima)
NewForecast2
```

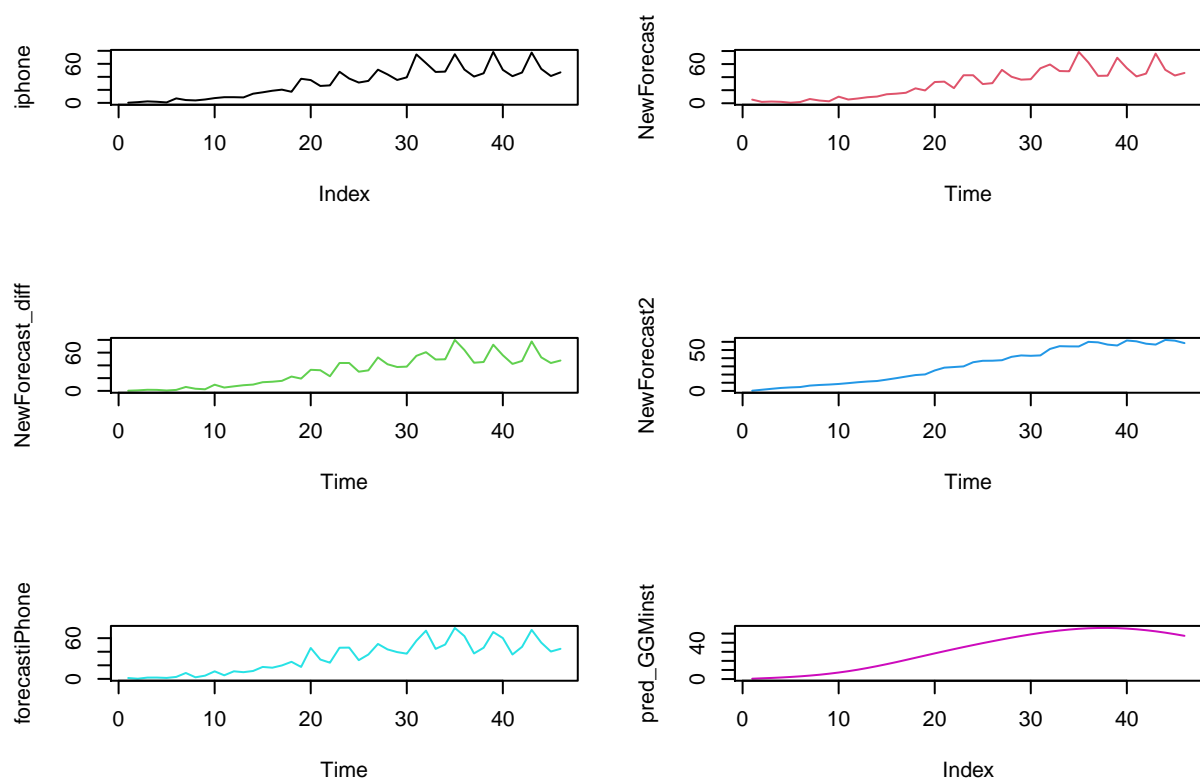
```
## Time Series:
## Start = 1
## End = 46
## Frequency = 1
## [1]  0.2710037  1.4529066  2.5713716  3.6585594  4.3149428  4.7748408
## [7]  6.5053003  7.2793961  7.7807721  8.4958783  9.5263741 10.6298322
## [13] 11.5000542 12.1131851 13.8108814 15.6024453 17.5253456 19.3981451
## [19] 20.1767839 25.0402197 28.4465936 29.2058838 29.9909851 35.0529991
## [25] 36.8325759 36.9159358 37.5264323 41.6741525 43.3832515 42.9152332
## [31] 43.4130721 51.2967667 54.6718138 54.4254526 54.3422010 59.9657637
## [37] 59.3716400 56.6074710 55.5192056 61.6393126 60.5974790 57.7064944
## [43] 56.6333353 62.3098653 61.4360542 58.4240565
```

```
#Plot comparison
```

```
plot(iphone, type='l', lty=1, col=1)
lines(NewForecast, lty=1, col=2)
lines(NewForecast_diff, lty=1, col=3)
lines(NewForecast2, lty=1, col=4)
lines(forecastiPhone, col=5)
lines(pred_GGMinst,col=6)
legend(0,81,legend= c("observed t.s.", "direct manual ARIMA", "direct manual ARIMA_diff", "direct auto
```



```
#Another view of results
par(mfrow=c(3,2))
plot(iphone, type='l', lty=1, col=1)
plot(NewForecast, type='l', lty=1, col=2)
plot(NewForecast_diff, type='l', lty=1, col=3)
plot(NewForecast2, type='l', lty=1, col=4)
plot(forecastiPhone, type='l', lty=1, col=5)
plot(pred_GGMinst, type='l', lty=1, col=6)
```

```
par(mfrow=c(1,1))
```

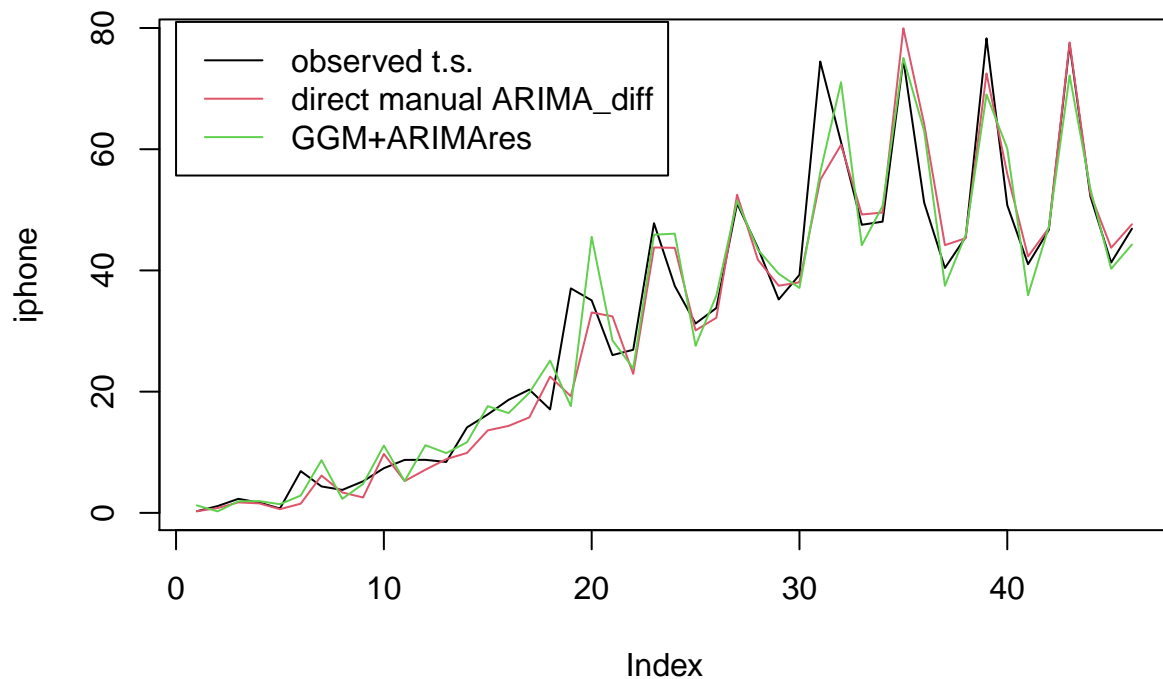
```
#Comparison between the best models: direct ARIMA with d=1 and GGM+ARIMA on residuals
```

```
plot(iphone, type='l', lty=1, col=1)
```

```
lines(NewForecast_diff, lty=1, col=2)
```

```
lines(forecastiPhone, col=3)
```

```
legend(0,81,legend= c("observed t.s.", "direct manual ARIMA_diff", "GGM+ARIMAs"), col=c(1,2,3), lty=1,
```



#To compare not only with plots, we can plot the behavior of residuals of our two best model

#Residual of direct manual ARIMA_diff

```
residuals_arima_diff <- residuals(directarima_diff)
residuals_arima_diff
```

Time Series:

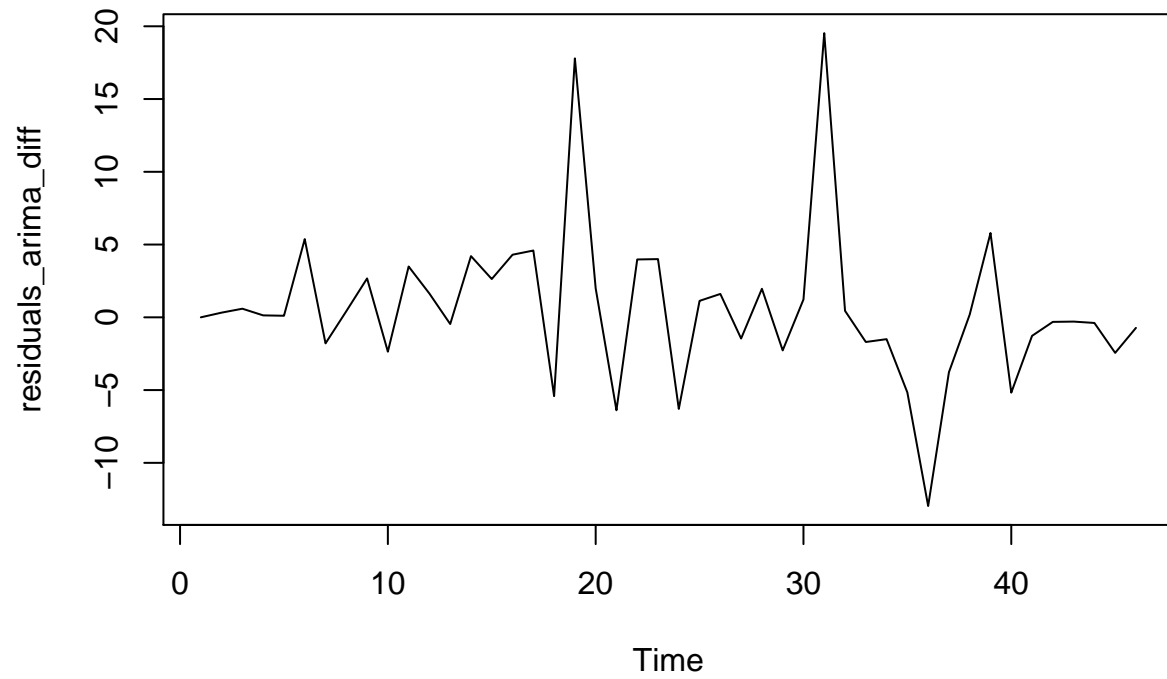
Start = 1

End = 46

Frequency = 1

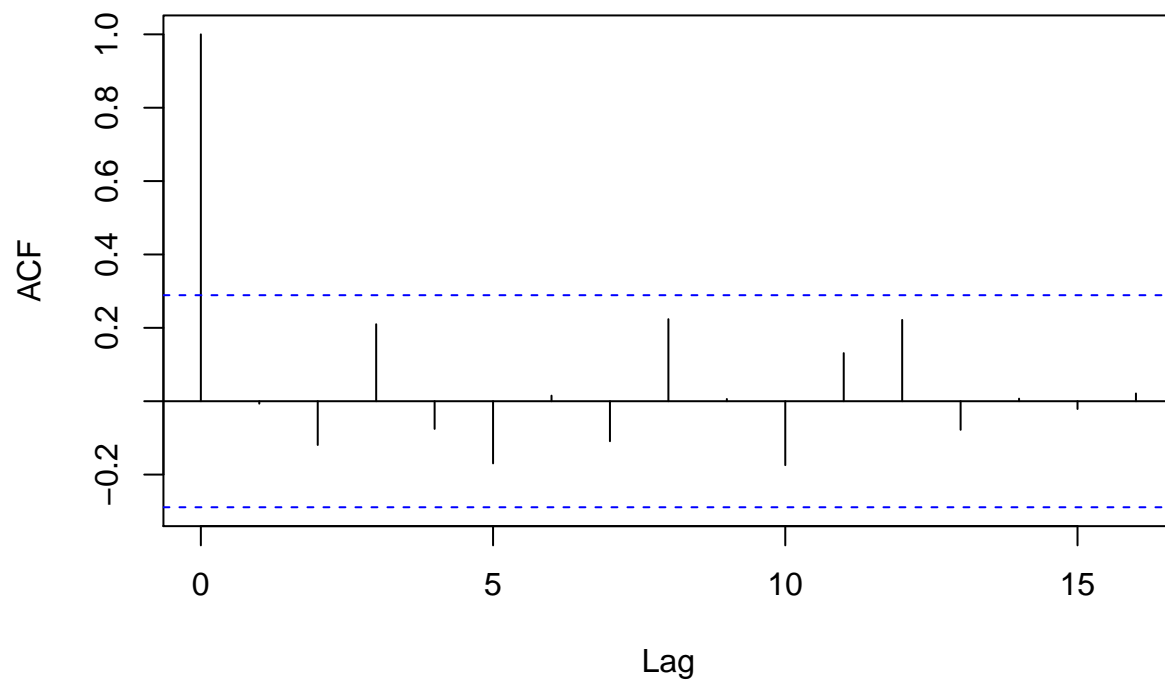
```
## [1] 2.699991e-04 3.214397e-01 5.913016e-01 1.330400e-01 1.077036e-01
## [6] 5.374082e+00 -1.790555e+00 4.184684e-01 2.673178e+00 -2.361293e+00
## [11] 3.492002e+00 1.630839e+00 -4.622558e-01 4.207926e+00 2.631406e+00
## [16] 4.301990e+00 4.590641e+00 -5.418043e+00 1.779489e+01 1.990081e+00
## [21] -6.386589e+00 3.976377e+00 3.999738e+00 -6.294503e+00 1.132315e+00
## [26] 1.606497e+00 -1.456622e+00 1.962006e+00 -2.273535e+00 1.226427e+00
## [31] 1.953036e+01 4.461971e-01 -1.695683e+00 -1.500910e+00 -5.164264e+00
## [36] -1.296689e+01 -3.772028e+00 1.879389e-01 5.794565e+00 -5.175036e+00
## [41] -1.273149e+00 -3.141701e-01 -2.929730e-01 -3.866012e-01 -2.442602e+00
## [46] -7.264280e-01
```

```
plot(residuals_arima_diff, type= 'l')
```



```
acf(residuals_arima_diff)
```

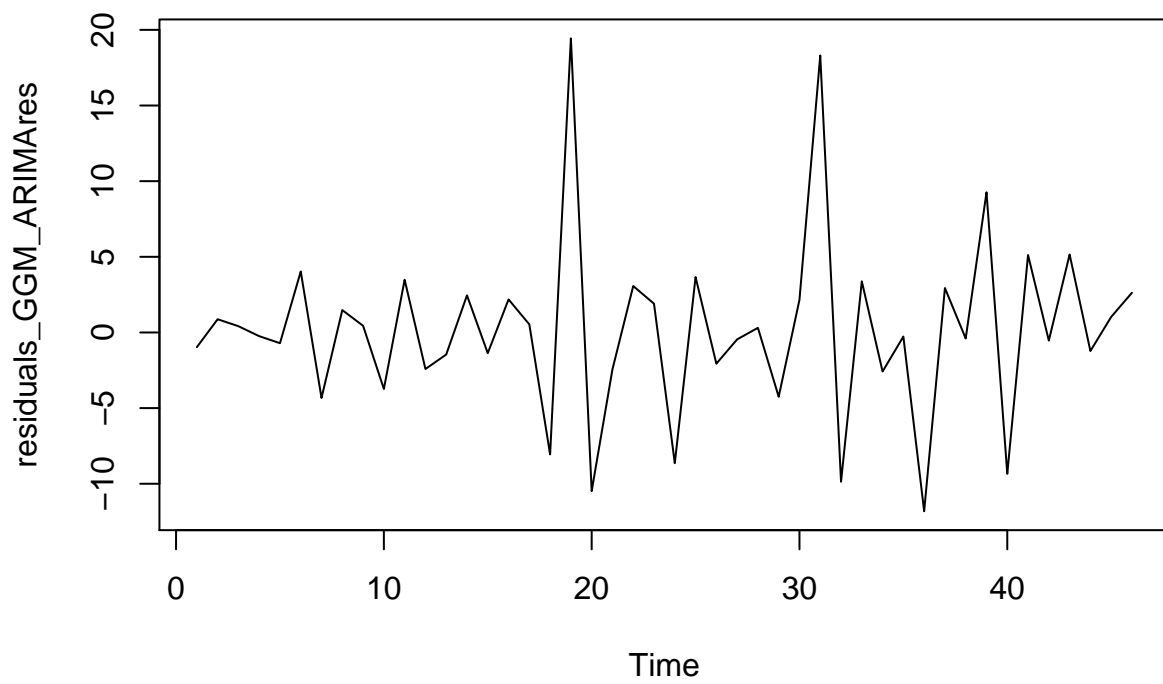
Series residuals_arima_diff



```
#Residual of GGM+ARIMAs
residuals_GGM_ARIMAs <- iphone - forecastiPhone
residuals_GGM_ARIMAs
```

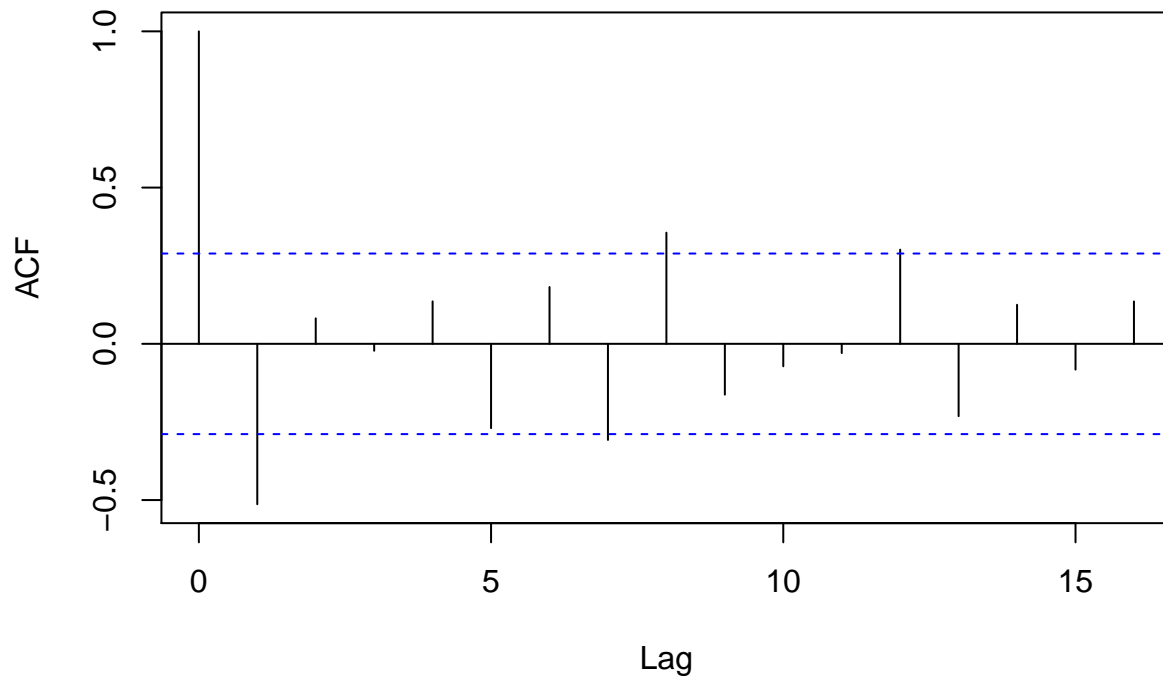
```
## Time Series:
## Start = 1
## End = 46
## Frequency = 1
## [1] -0.9724115  0.8736653  0.4146908 -0.2440406 -0.7089084  4.0283915
## [7] -4.3278219  1.4794408  0.4421643 -3.7354608  3.4833033 -2.4159624
## [13] -1.4590221  2.4519179 -1.3681216  2.1854937  0.5324046 -8.0609977
## [19] 19.4389341 -10.4865230 -2.4379123  3.0686162  1.9027494 -8.6403140
## [25]  3.6629811 -2.0628538 -0.4565629  0.3052399 -4.2509579  2.1576391
## [31] 18.3079676 -9.8676161  3.3772768 -2.5813194 -0.2702617 -11.8158616
## [37]  2.9360662 -0.3998982  9.2635014 -9.3429145  5.1190534 -0.5360798
## [43]  5.1536433 -1.2258086  1.0272065  2.6246228
```

```
plot(residuals_GGM_ARIMAs, type= 'l')
```



```
acf(residuals_GGM_ARIMAs)
```

Series residuals_GGM_ARIMAs

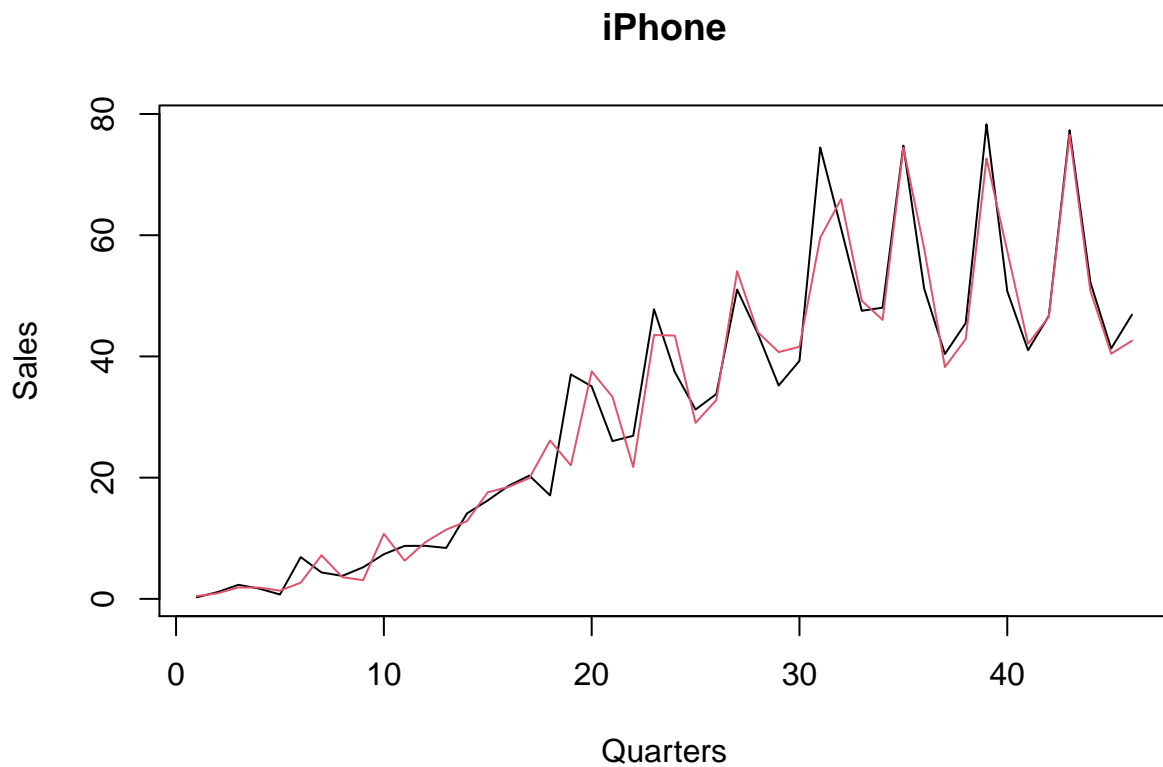


```
#ARMAX refinement (ARIMA model with x regressor - predicted values with GGM)
arimaGGM<-Arima(iphonec, order = c(2,0,1),seasonal = list(order=c(1,0,0), period=4), xreg = pred_GGM)
arimaGGM
```

```
## Series: iphonec
## Regression with ARIMA(2,0,1)(1,0,0)[4] errors
##
## Coefficients:
##          ar1          ar2          ma1          sar1  intercept          xreg
##          1.6394         -0.9062        -1.0000         0.913           0.3183         1.0003
## s.e.    0.0588         0.0553         0.0295         0.052           0.6622         0.0012
##
## sigma^2 estimated as 12.44:  log likelihood=-123.83
## AIC=261.66   AICc=264.6   BIC=274.46
```

```
#Predictions
pred_sarimax<- fitted(arimaGGM)
pred_sarmaxinst<- make.instantaneous(fitted(arimaGGM))
```

```
#Plot
plot(iphone, type= "l",xlab="Quarters", ylab="Sales", main="iPhone")
lines(pred_sarmaxinst,col=2)
```



LAB 2

```
library("readxl")
#data from Athanaspoulos and Hyndman
library(fpp2)
#DW test
library(lmtest)
library(forecast)
```

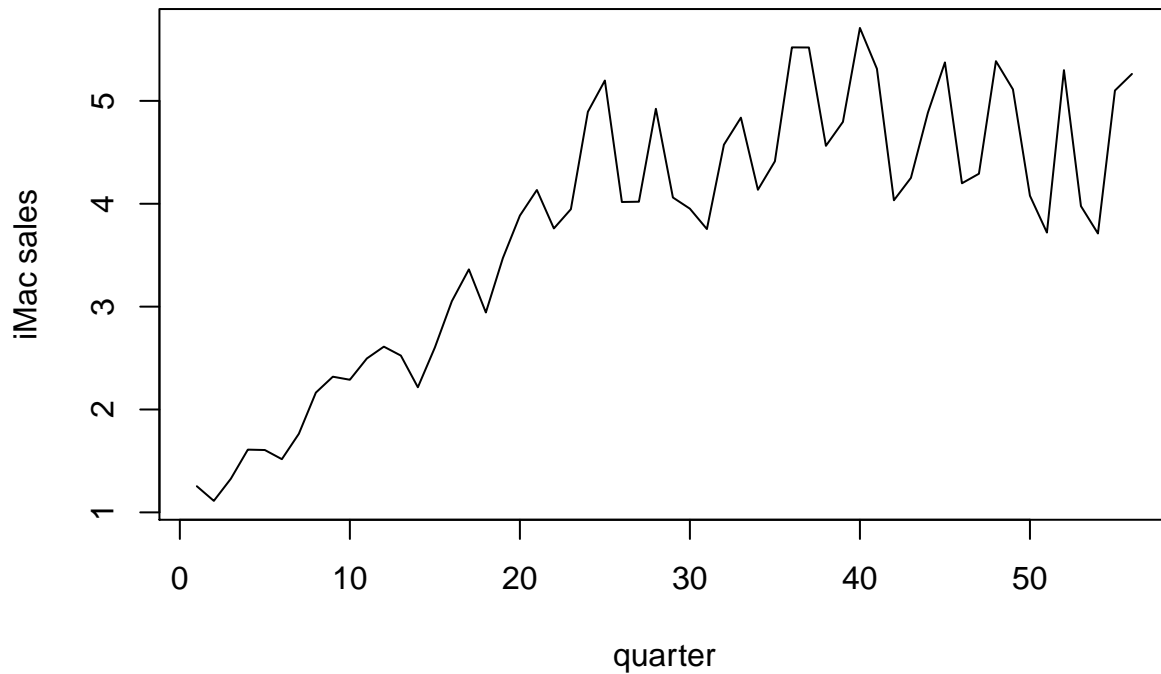
LINEAR REGRESSION

```
#iMac example
apple<- read_excel("DatiAPPLE.xlsx")
str(apple)
```

```
## tibble [56 x 4] (S3: tbl_df/tbl/data.frame)
## $ iPhone: num [1:56] 0.27 1.12 2.32 1.7 0.72 6.89 4.36 3.79 5.21 7.37 ...
## $ iPad : num [1:56] 3.27 4.19 7.33 4.69 9.25 ...
## $ iPod : num [1:56] 14.04 8.53 8.11 8.73 21.07 ...
## $ iMac : num [1:56] 1.25 1.11 1.33 1.61 1.61 ...
```

```
imac <- apple$iMac
```

```
#Data visualization  
plot(imac,type="l", xlab="quarter", ylab="iMac sales")
```



```
#Time variable "tt" for a linear model  
tt<- 1:NROW(apple)
```

```
#Linear model  
fitlm <- lm(imac~tt)  
summary(fitlm)
```

```
##  
## Call:  
## lm(formula = imac ~ tt)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -1.70990 -0.57455  0.01084  0.41485  1.66010   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept)  1.914625   0.200379   9.555 3.34e-13 ***  
## tt           0.064931   0.006116  10.617 7.87e-15 ***
```



```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7397 on 54 degrees of freedom
## Multiple R-squared:  0.6761, Adjusted R-squared:  0.6701
## F-statistic: 112.7 on 1 and 54 DF,  p-value: 7.871e-15
```

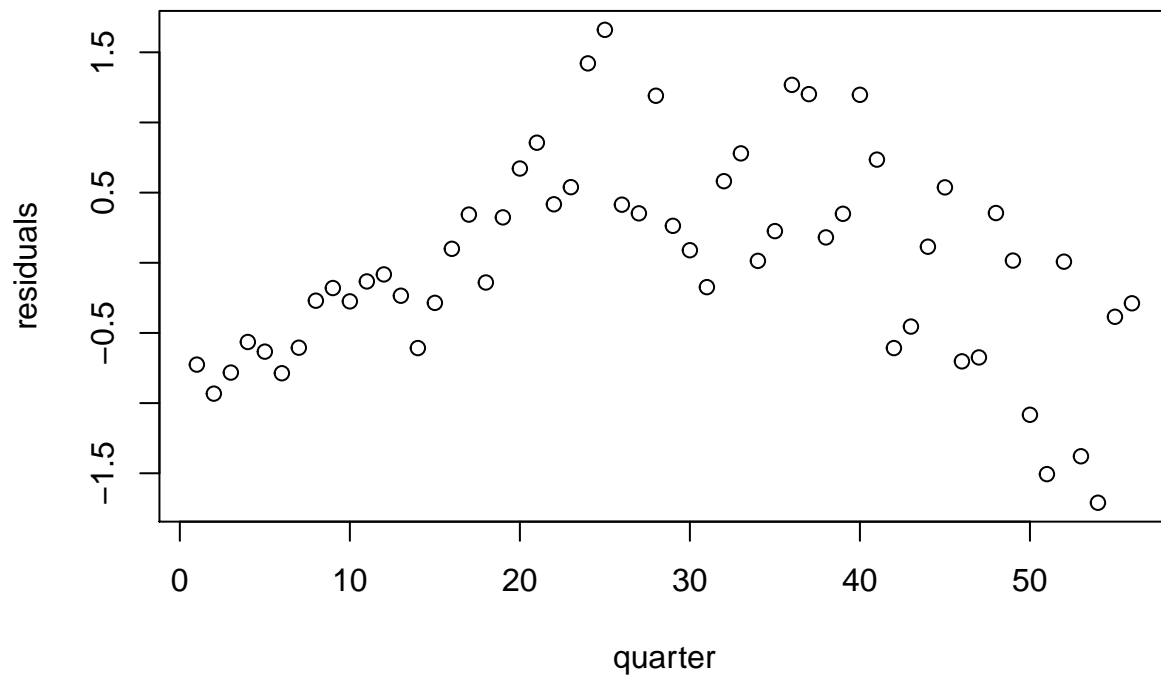
```
#Residuals
```

```
res<- residuals(fitlm)
```

```
#Plot od residuals
```

```
#Parabolic behavior, they aren't randomly distributed
```

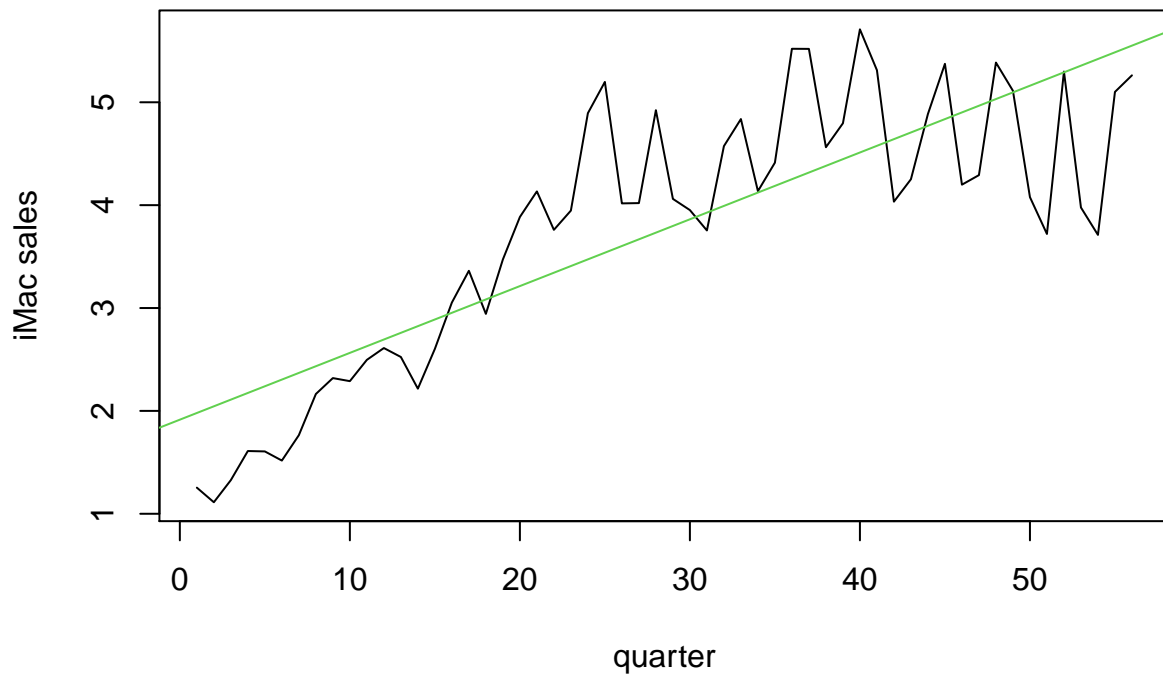
```
plot(res, xlab="quarter", ylab="residuals")
```



```
#Plot of the model
```

```
plot(imac,type="l", xlab="quarter", ylab="iMac sales")
```

```
abline(fitlm, col=3)
```



```
#Data transformed as time series
mac.ts<-ts(imac, frequency=4)
mac.ts
```

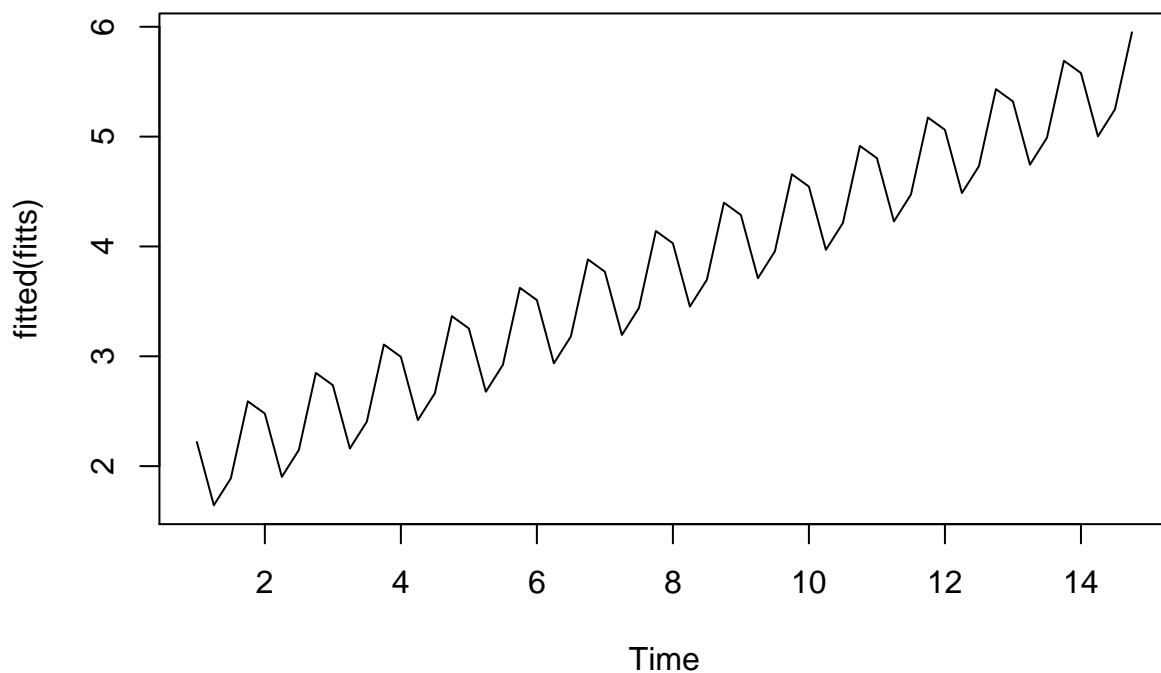
```
##      Qtr1  Qtr2  Qtr3  Qtr4
## 1  1.254  1.112  1.327  1.610
## 2  1.606  1.517  1.764  2.164
## 3  2.319  2.289  2.496  2.611
## 4  2.524  2.216  2.603  3.053
## 5  3.362  2.943  3.472  3.885
## 6  4.134  3.760  3.947  4.894
## 7  5.198  4.017  4.020  4.923
## 8  4.061  3.952  3.754  4.574
## 9  4.837  4.136  4.413  5.520
## 10 5.519  4.563  4.796  5.709
## 11 5.312  4.034  4.252  4.886
## 12 5.374  4.199  4.292  5.386
## 13 5.112  4.078  3.720  5.299
## 14 3.977  3.711  5.101  5.262
```

```
#Model with trend and seasonality
#"trend" and "seasonality" are two variables inside the tslm function
fitts <- tslm(mac.ts~ trend+season)
#Season1 is omitted because the three season (2,3,4) are calculated as the difference from
#a "base state", in this case season1. So we avoid "perfect-correlation"
summary(fitts)
```

```
##
## Call:
## tslm(formula = mac.ts ~ trend + season)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.60158 -0.42293 -0.00687  0.54972  1.42797
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.155255   0.236078   9.129 2.62e-12 ***
## trend        0.064591   0.005613  11.507 8.68e-16 ***
## season2     -0.640448   0.256052  -2.501  0.0156 *
## season3     -0.460039   0.256237  -1.795  0.0785 .
## season4      0.176727   0.256544   0.689  0.4940
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6773 on 51 degrees of freedom
## Multiple R-squared:  0.7436, Adjusted R-squared:  0.7235
## F-statistic: 36.97 on 4 and 51 DF,  p-value: 1.695e-14
```

```
rests <- residuals(fitts)
```

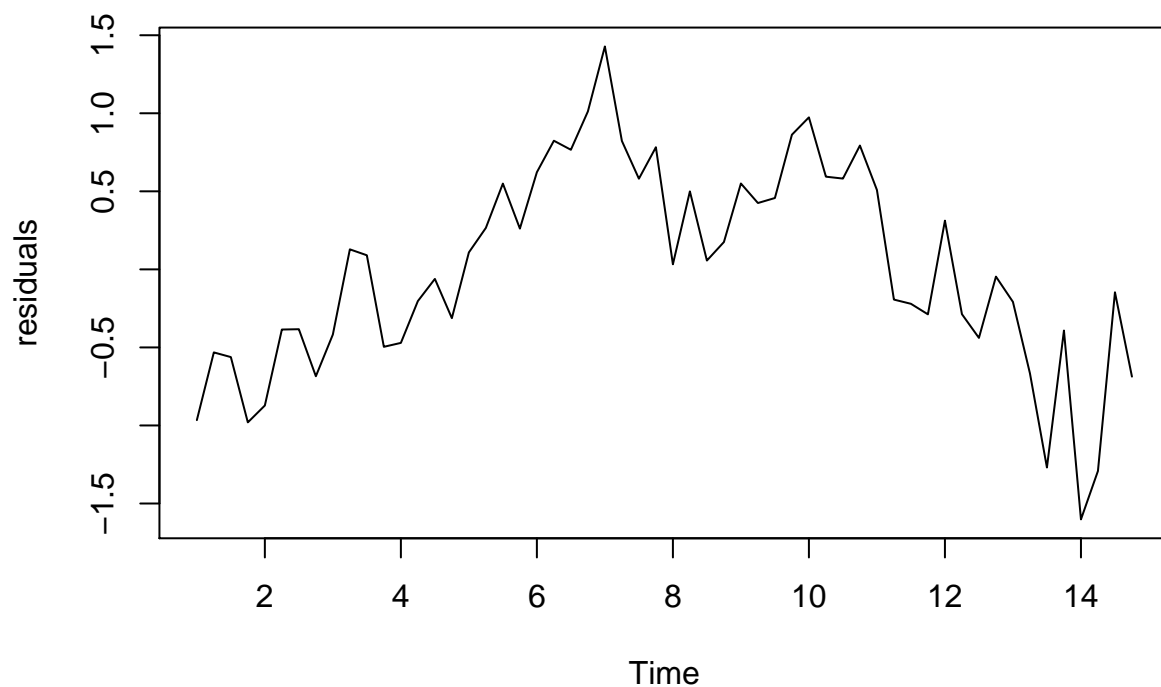
```
#Plot
plot(fitted(fitts))
```



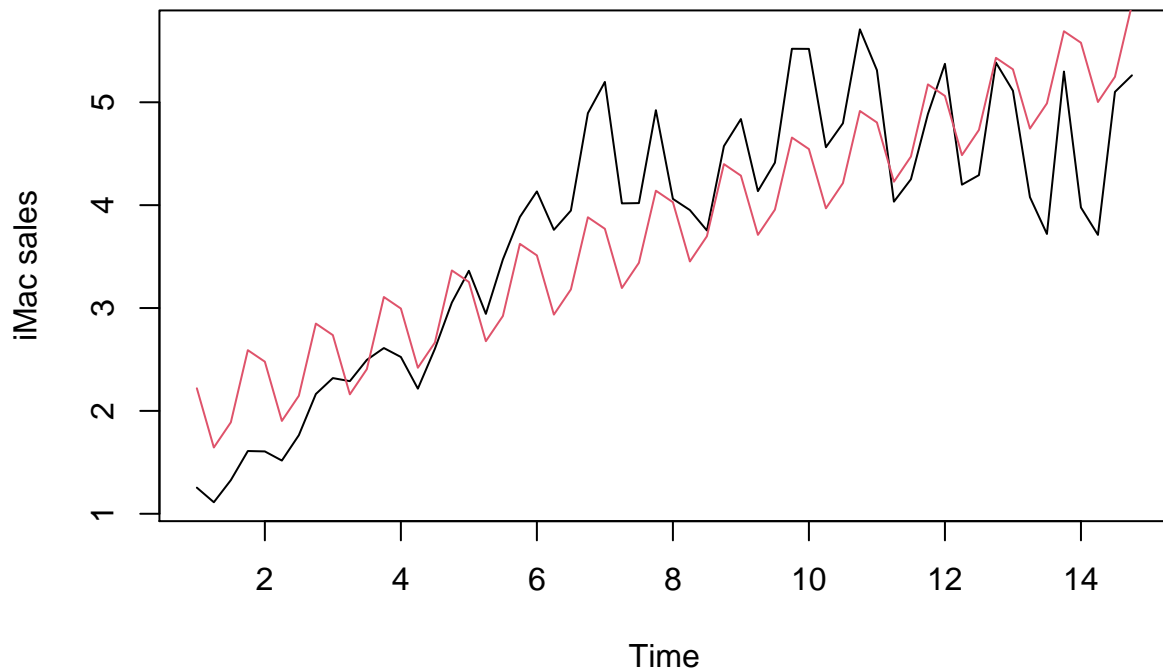
```
#Durbin-Watson test
#DW value is between 0 and 4 (2 the middle value). If it is near zero it means that the residuals
#are autocorrelated (see also p-value).
dwtest(fitts)
```

```
##
## Durbin-Watson test
##
## data: fitts
## DW = 0.44182, p-value = 5.722e-13
## alternative hypothesis: true autocorrelation is greater than 0
```

```
#Plt of residuals
plot(rests, ylab="residuals")
```



```
#Plot of model
plot(mac.ts, ylab="iMac sales", xlab="Time")
lines(fitted(fitts), col=2)
```

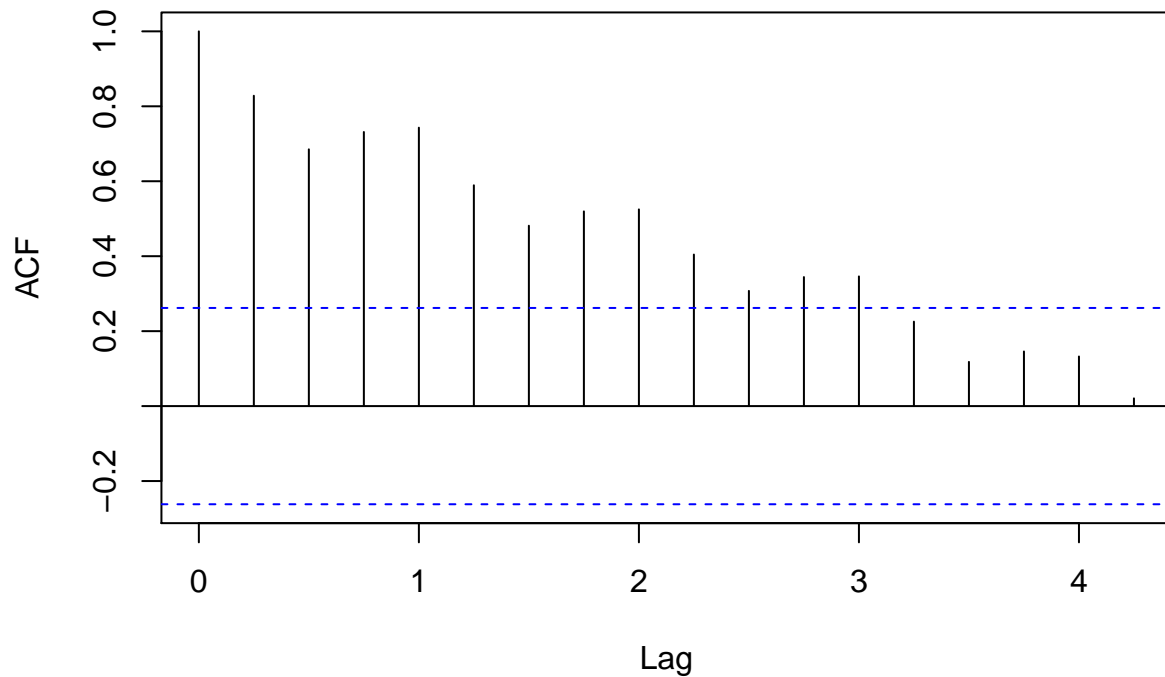


ARIMA MODEL

The graph of autocorrelations shows an increasing trend: decreasing behavior of autocorrelations but significant for many lags, so this indicates a trend effect of seasonality demonstrated by the fact that we have peaks every 4 lag. It is possible to use an ARIMA model with trend and seasonality

```
acf(mac.ts)
```

Series mac.ts



#The two vector are obtained by trials and errors

```
arima<- Arima(mac.ts, order = c(0,1,1), seasonal = list(order=c(0,1,1), period=4))
arima
```

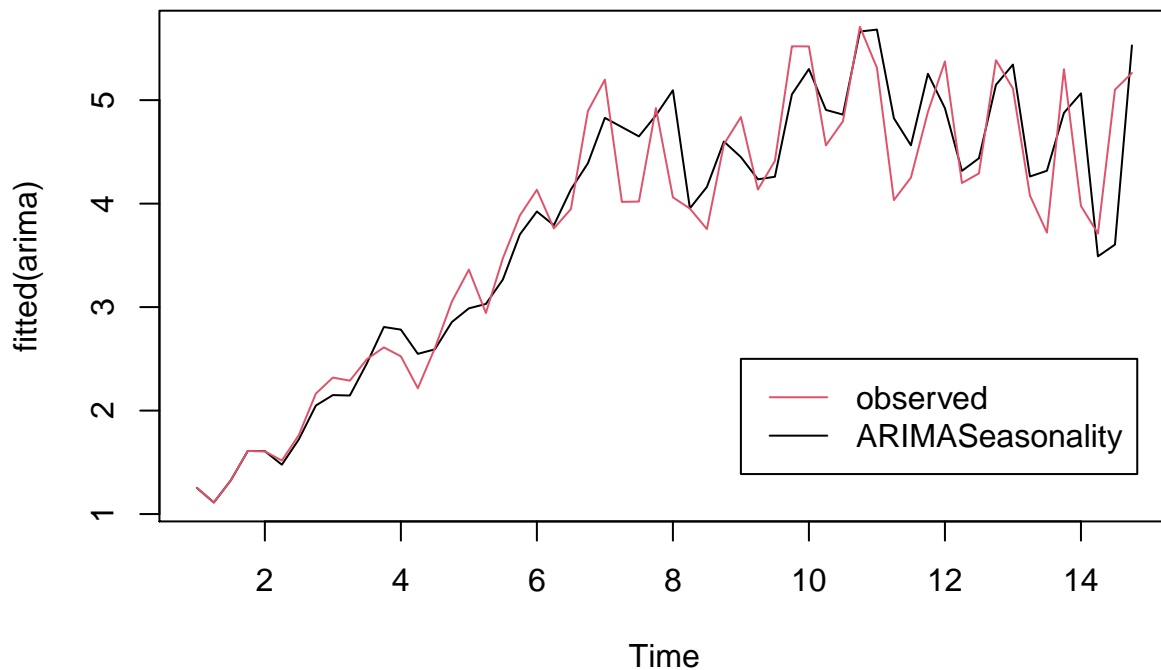
```
## Series: mac.ts
## ARIMA(0,1,1)(0,1,1)[4]
##
## Coefficients:
##          ma1      sma1
##      -0.4534  -0.6822
## s.e.   0.1298   0.1159
##
## sigma^2 estimated as 0.1879: log likelihood=-30.11
## AIC=66.22   AICc=66.74   BIC=72.02
```

```
fitted(arima)
```

```
##          Qtr1      Qtr2      Qtr3      Qtr4
## 1  1.253276  1.111786  1.326678  1.609974
## 2  1.608018  1.477129  1.721834  2.048957
## 3  2.149640  2.145573  2.455912  2.807907
## 4  2.782325  2.548208  2.592214  2.855874
## 5  2.987849  3.030162  3.266481  3.702632
## 6  3.923661  3.790895  4.136874  4.389501
## 7  4.828762  4.739445  4.649783  4.852963
```

```
## 8  5.095361 3.954345 4.161322 4.599973
## 9  4.449660 4.234780 4.259724 5.055608
## 10 5.300484 4.905838 4.860421 5.662982
## 11 5.681540 4.824825 4.563490 5.254897
## 12 4.922537 4.316473 4.438149 5.147580
## 13 5.343817 4.261874 4.317721 4.877185
## 14 5.065589 3.490321 3.603849 5.528511
```

```
plot(fitted(arima))
lines(mac.ts, col=2)
legend(9,2.5,legend= c("observed","ARIMASeasonality"), col=c(2,1), lty=1)
```



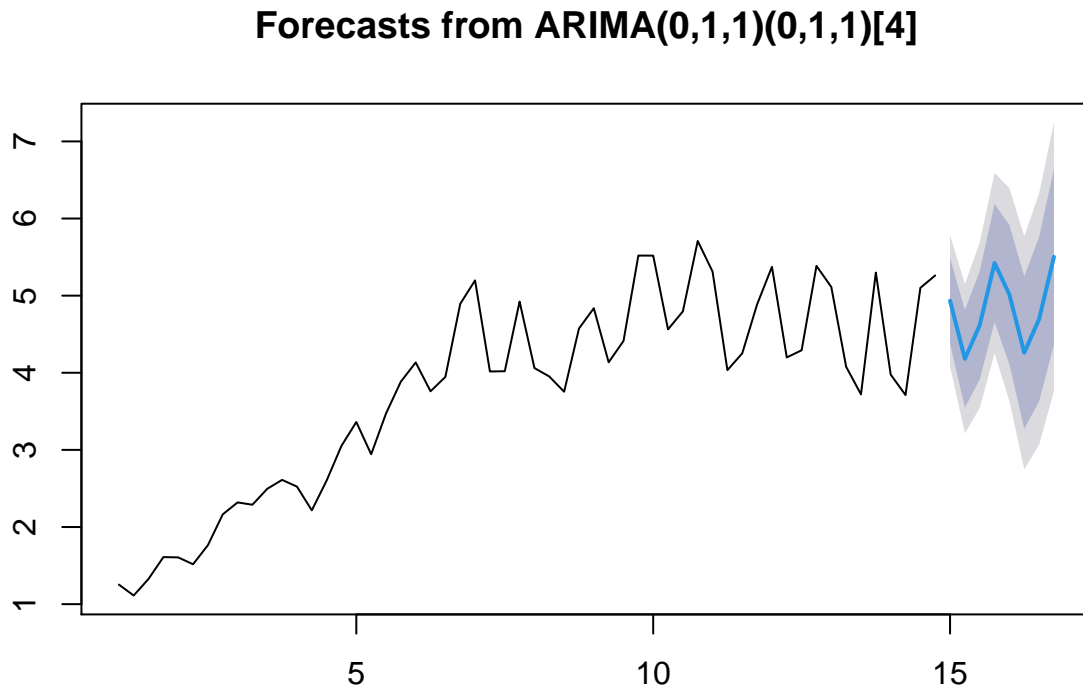
Pro: we have a very good fit (good forecasting) Cons: we can't interpret in a good way the meaning of our parameters (the two vectors of ARIMA) (bad interpretability)

```
#Forecasting (with confidence intervals)
forecast(arima)
```

##	Point	Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 15	Q1	4.933958	4.378410	5.489506	4.084321	5.783596
## 15	Q2	4.180688	3.547570	4.813806	3.212418	5.148959
## 15	Q3	4.617581	3.915410	5.319751	3.543704	5.691458
## 15	Q4	5.423500	4.658484	6.188515	4.253510	6.593489
## 16	Q1	5.013020	4.109776	5.916264	3.631628	6.394412
## 16	Q2	4.259750	3.271838	5.247661	2.748869	5.770630

```
## 16 Q3      4.696642 3.630767 5.762517 3.066527 6.326758
## 16 Q4      5.502561 4.364049 6.641073 3.761357 7.243765
```

```
plot(forecast(arima))
```

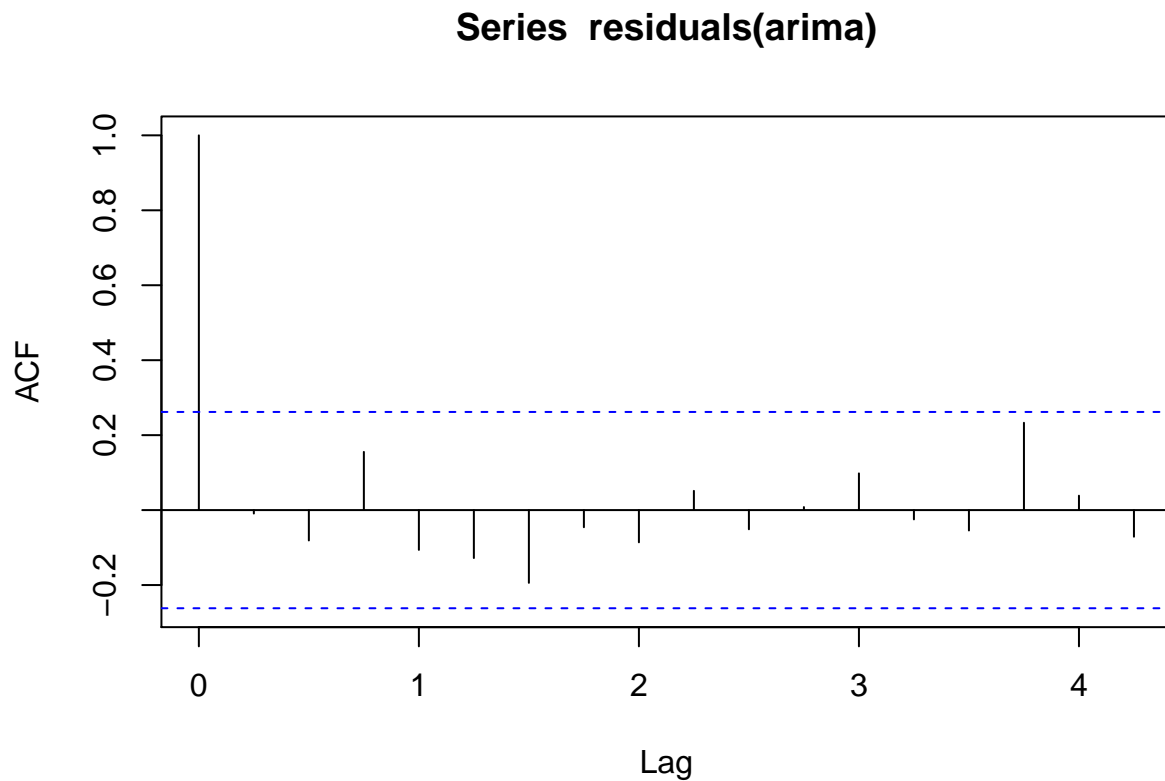


The acf of residuals is very satisfactory (aside from the first autocorrelation) we will deal with a residual situation with constant mean(=0) and constant variance.

```
#We have white noise in residuals
residuals(arima)
```

##	Qtr1	Qtr2	Qtr3	Qtr4
## 1	7.239970e-04	2.137886e-04	3.216655e-04	2.552349e-05
## 2	-2.017717e-03	3.987147e-02	4.216618e-02	1.150428e-01
## 3	1.693596e-01	1.434271e-01	4.008797e-02	-1.969067e-01
## 4	-2.583252e-01	-3.322081e-01	1.078648e-02	1.971261e-01
## 5	3.741511e-01	-8.716242e-02	2.055195e-01	1.823675e-01
## 6	2.103394e-01	-3.089534e-02	-1.898741e-01	5.044994e-01
## 7	3.692375e-01	-7.224451e-01	-6.297831e-01	7.003665e-02
## 8	-1.034361e+00	-2.345378e-03	-4.073217e-01	-2.597308e-02
## 9	3.873404e-01	-9.878010e-02	1.532763e-01	4.643922e-01
## 10	2.185157e-01	-3.428377e-01	-6.442051e-02	4.601807e-02
## 11	-3.695397e-01	-7.908254e-01	-3.114898e-01	-3.688967e-01
## 12	4.514633e-01	-1.174731e-01	-1.461493e-01	2.384195e-01
## 13	-2.318174e-01	-1.838741e-01	-5.977206e-01	4.218152e-01
## 14	-1.088589e+00	2.206788e-01	1.497151e+00	-2.665114e-01


```
acf(residuals(arima))
```



```
#auto ARIMA  
#The auto.arima shows the same results  
auto.arima(mac.ts)
```

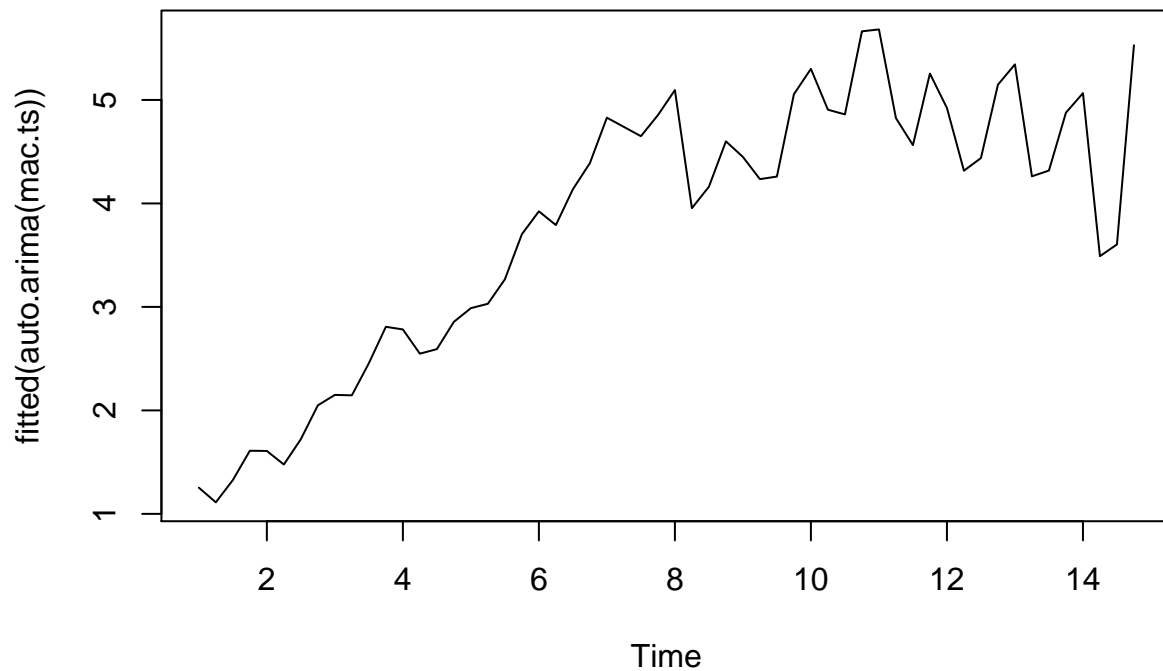
```
## Series: mac.ts  
## ARIMA(0,1,1)(0,1,1)[4]  
##  
## Coefficients:  
##          ma1      sma1  
##      -0.4534 -0.6822  
## s.e.   0.1298   0.1159  
##  
## sigma^2 estimated as 0.1879: log likelihood=-30.11  
## AIC=66.22   AICc=66.74   BIC=72.02
```

```
fitted(auto.arima(mac.ts))
```

```
##      Qtr1      Qtr2      Qtr3      Qtr4  
## 1  1.253276 1.111786 1.326678 1.609974  
## 2  1.608018 1.477129 1.721834 2.048957  
## 3  2.149640 2.145573 2.455912 2.807907  
## 4  2.782325 2.548208 2.592214 2.855874
```

```
## 5 2.987849 3.030162 3.266481 3.702632
## 6 3.923661 3.790895 4.136874 4.389501
## 7 4.828762 4.739445 4.649783 4.852963
## 8 5.095361 3.954345 4.161322 4.599973
## 9 4.449660 4.234780 4.259724 5.055608
## 10 5.300484 4.905838 4.860421 5.662982
## 11 5.681540 4.824825 4.563490 5.254897
## 12 4.922537 4.316473 4.438149 5.147580
## 13 5.343817 4.261874 4.317721 4.877185
## 14 5.065589 3.490321 3.603849 5.528511
```

```
#Plot
#plot(auto.arima(mac.ts))
plot(fitted(auto.arima(mac.ts)))
```

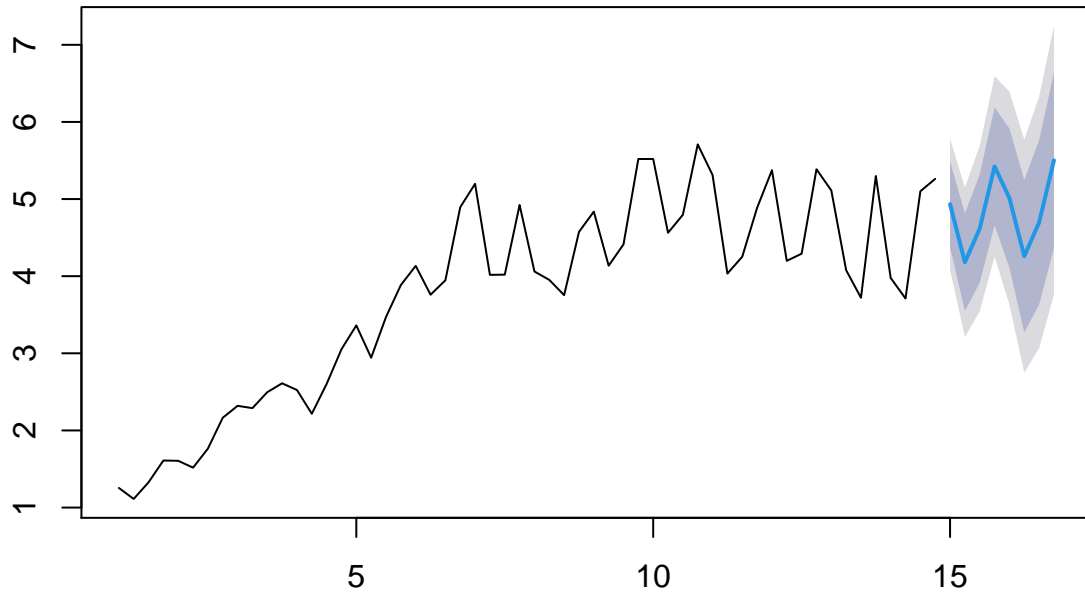


```
forecast(auto.arima(mac.ts))
```

##	Point	Forecast	Lo 80	Hi 80	Lo 95	Hi 95
## 15	Q1	4.933958	4.378410	5.489506	4.084321	5.783596
## 15	Q2	4.180688	3.547570	4.813806	3.212418	5.148959
## 15	Q3	4.617581	3.915410	5.319751	3.543704	5.691458
## 15	Q4	5.423500	4.658484	6.188515	4.253510	6.593489
## 16	Q1	5.013020	4.109776	5.916264	3.631628	6.394412
## 16	Q2	4.259750	3.271838	5.247661	2.748869	5.770630
## 16	Q3	4.696642	3.630767	5.762517	3.066527	6.326758
## 16	Q4	5.502561	4.364049	6.641073	3.761357	7.243765

```
plot(forecast(auto.arima(mac.ts)))
```

Forecasts from ARIMA(0,1,1)(0,1,1)[4]



Antidiabetic drugs example

```
a10
```

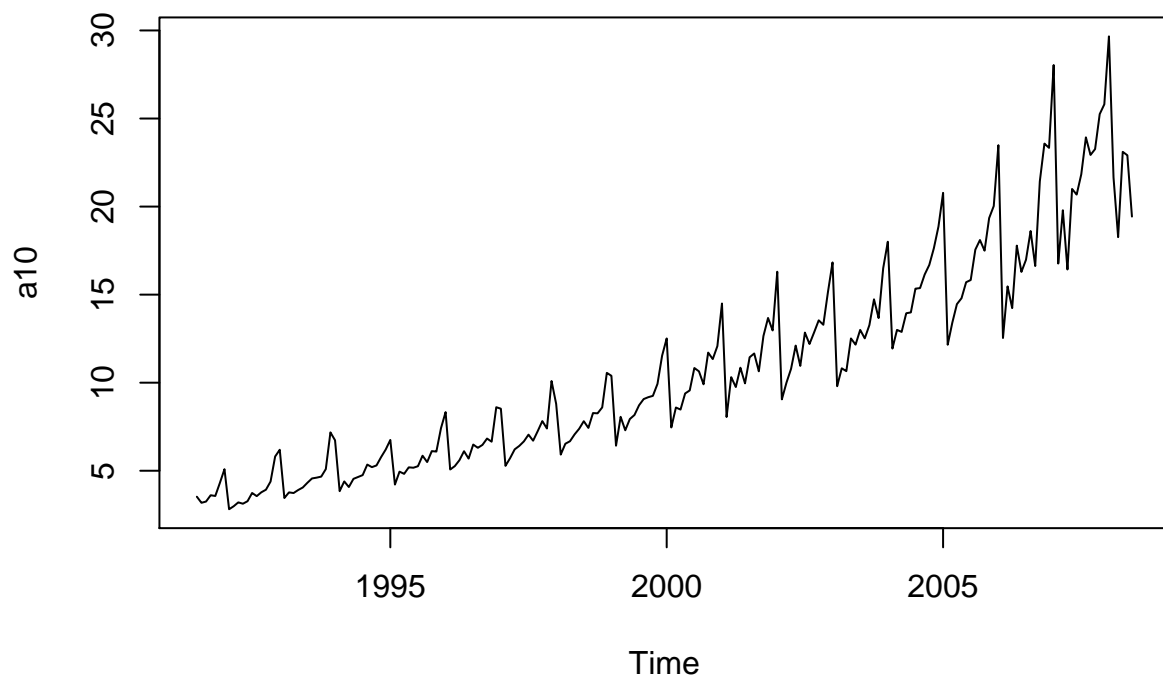
##	Jan	Feb	Mar	Apr	May	Jun	Jul
## 1991							3.526591
## 1992	5.088335	2.814520	2.985811	3.204780	3.127578	3.270523	3.737851
## 1993	6.192068	3.450857	3.772307	3.734303	3.905399	4.049687	4.315566
## 1994	6.731473	3.841278	4.394076	4.075341	4.540645	4.645615	4.752607
## 1995	6.749484	4.216067	4.949349	4.823045	5.194754	5.170787	5.256742
## 1996	8.329452	5.069796	5.262557	5.597126	6.110296	5.689161	6.486849
## 1997	8.524471	5.277918	5.714303	6.214529	6.411929	6.667716	7.050831
## 1998	8.798513	5.918261	6.534493	6.675736	7.064201	7.383381	7.813496
## 1999	10.391416	6.421535	8.062619	7.297739	7.936916	8.165323	8.717420
## 2000	12.511462	7.457199	8.591191	8.474000	9.386803	9.560399	10.834295
## 2001	14.497581	8.049275	10.312891	9.753358	10.850382	9.961719	11.443601
## 2002	16.300269	9.053485	10.002449	10.788750	12.106705	10.954101	12.844566
## 2003	16.828350	9.800215	10.816994	10.654223	12.512323	12.161210	12.998046
## 2004	18.003768	11.938030	12.997900	12.882645	13.943447	13.989472	15.339097
## 2005	20.778723	12.154552	13.402392	14.459239	14.795102	15.705248	15.829550
## 2006	23.486694	12.536987	15.467018	14.233539	17.783058	16.291602	16.980282

```
## 2007 28.038383 16.763869 19.792754 16.427305 21.000742 20.681002 21.834890
## 2008 29.665356 21.654285 18.264945 23.107677 22.912510 19.431740
##           Aug           Sep           Oct           Nov           Dec
## 1991  3.180891  3.252221  3.611003  3.565869  4.306371
## 1992  3.558776  3.777202  3.924490  4.386531  5.810549
## 1993  4.562185  4.608662  4.667851  5.093841  7.179962
## 1994  5.350605  5.204455  5.301651  5.773742  6.204593
## 1995  5.855277  5.490729  6.115293  6.088473  7.416598
## 1996  6.300569  6.467476  6.828629  6.649078  8.606937
## 1997  6.704919  7.250988  7.819733  7.398101 10.096233
## 1998  7.431892  8.275117  8.260441  8.596156 10.558939
## 1999  9.070964  9.177113  9.251887  9.933136 11.532974
## 2000 10.643751  9.908162 11.710041 11.340151 12.079132
## 2001 11.659239 10.647060 12.652134 13.674466 12.965735
## 2002 12.196500 12.854748 13.542004 13.287640 15.134918
## 2003 12.517276 13.268658 14.733622 13.669382 16.503966
## 2004 15.370764 16.142005 16.685754 17.636728 18.869325
## 2005 17.554701 18.100864 17.496668 19.347265 20.031291
## 2006 18.612189 16.623343 21.430241 23.575517 23.334206
## 2007 23.930204 22.930357 23.263340 25.250030 25.806090
## 2008
```

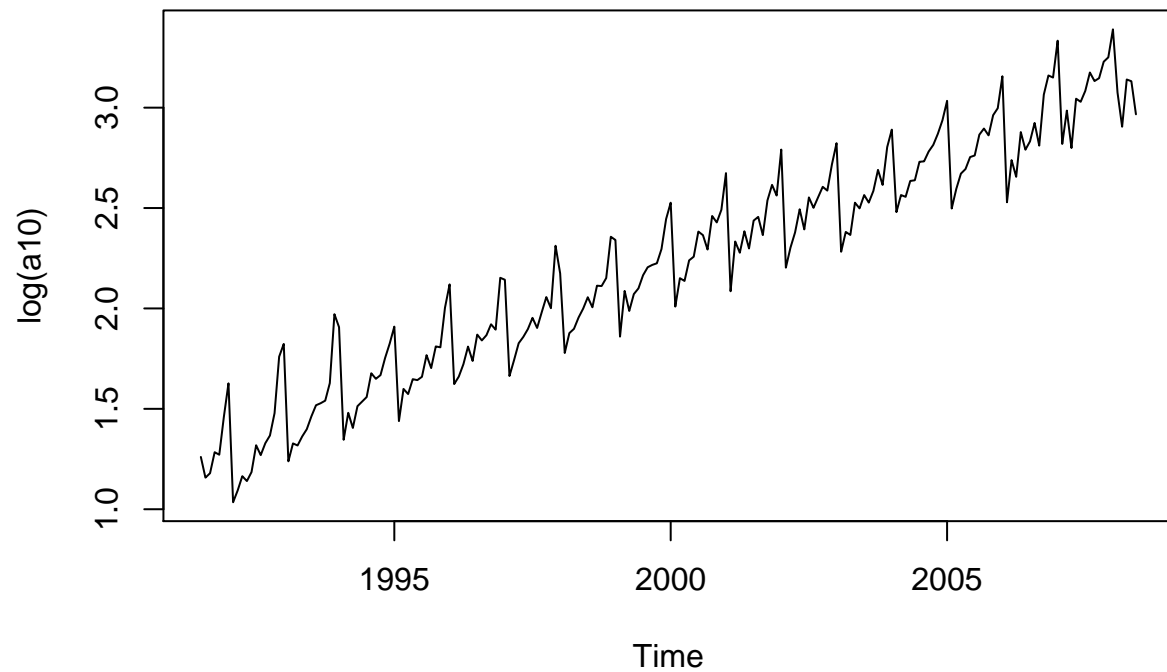
```
str(a10)
```

```
## Time-Series [1:204] from 1992 to 2008: 3.53 3.18 3.25 3.61 3.57 ...
```

```
#Plot
plot(a10)
```

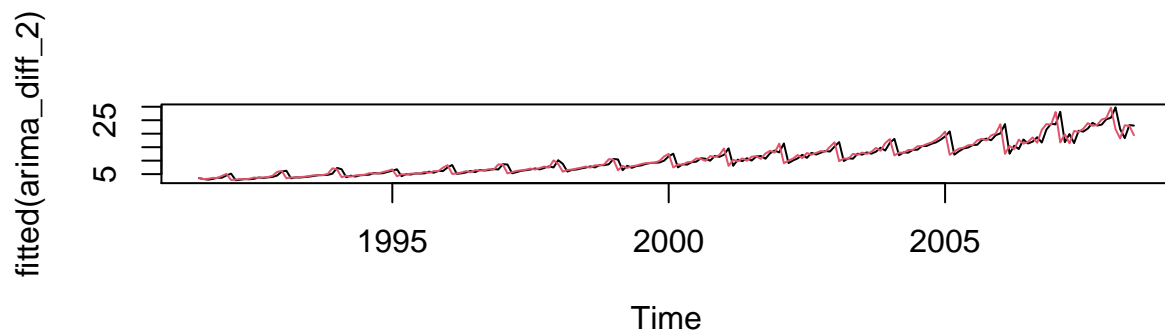
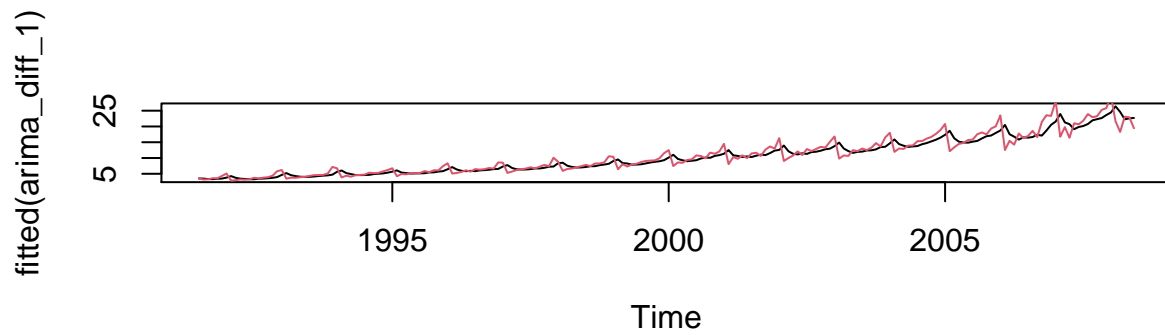


```
#Logaritimic transformation  
plot(log(a10))
```



###BEST ARIMA MODEL FOR ANTIDIABETIC DRUGS EXAMPLE

```
#EXERCISE (FIRST TRY)
par(mfrow=c(2,1))
arima_diff_1<- Arima(a10, order=c(0,1,1))
arima_diff_2<- Arima(a10, order=c(0,2,1))
plot(fitted(arima_diff_1))
lines(a10, col=2)
plot(fitted(arima_diff_2))
lines(a10, col=2)
```

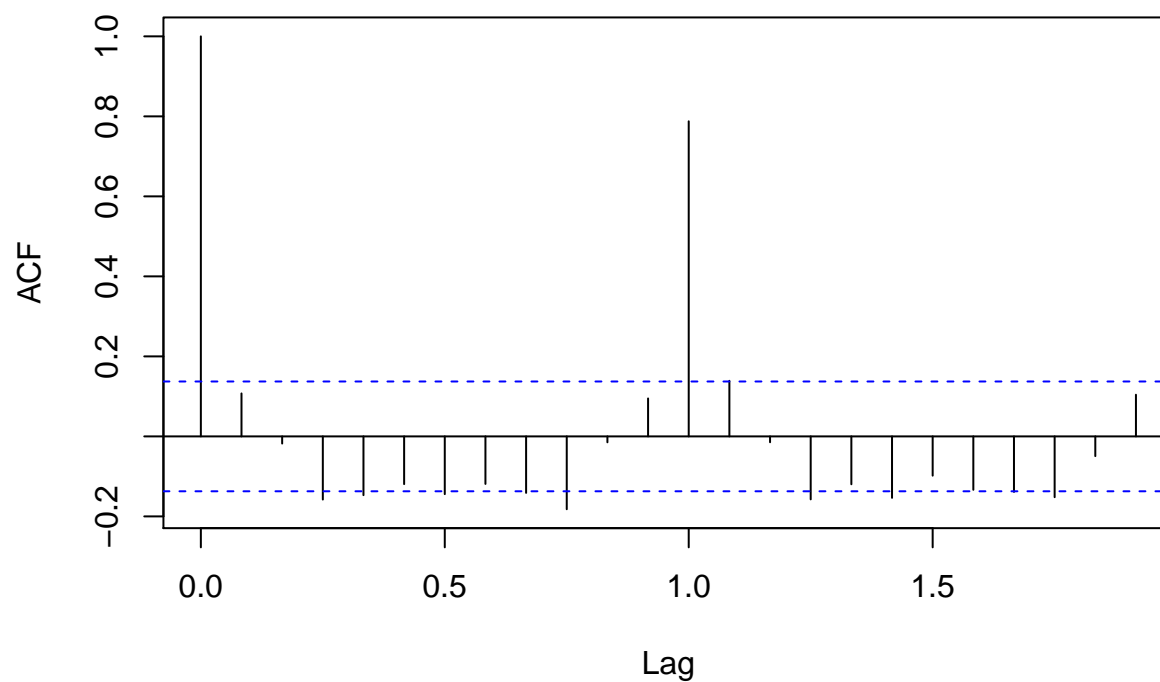


```
par(mfrow=c(1,1))
summary(arima_diff_1)
```

```
## Series: a10
## ARIMA(0,1,1)
##
## Coefficients:
##      ma1
##    -0.6289
## s.e.   0.1056
##
## sigma^2 estimated as 4.065:  log likelihood=-430.14
## AIC=864.28   AICc=864.34   BIC=870.91
##
## Training set error measures:
##           ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 0.238212 2.006291 1.377113 0.04481567 13.15865 1.063186 0.1074624
```

```
acf(residuals(arima_diff_1))
```

Series residuals(arima_diff_1)

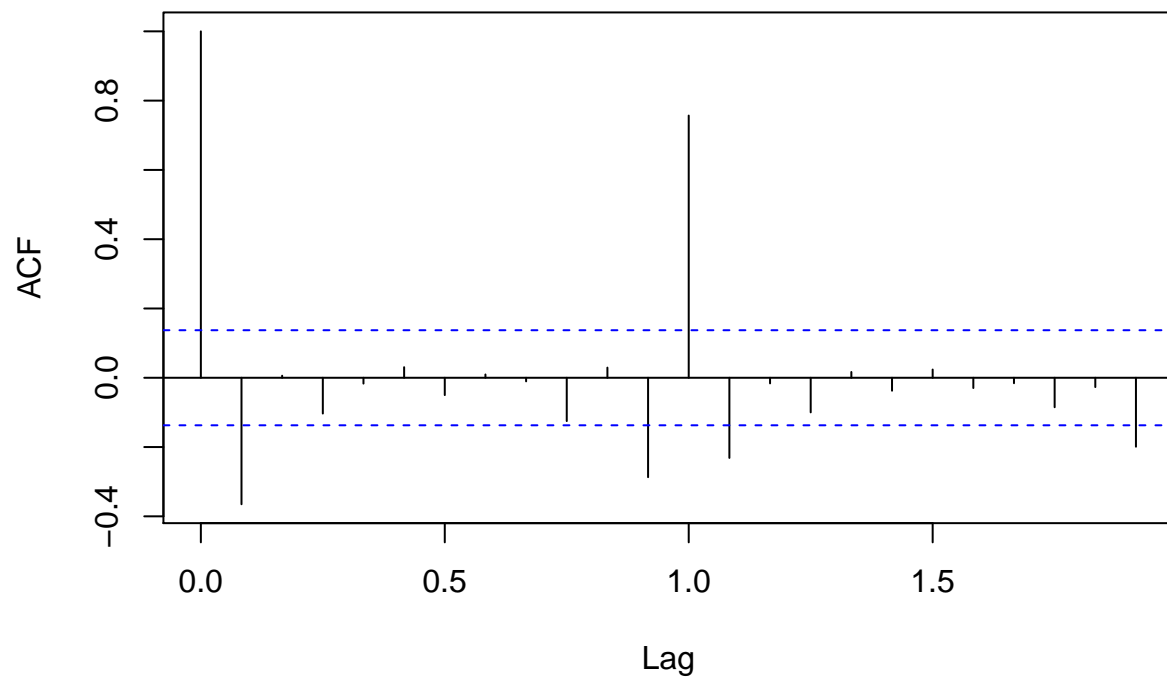


```
summary(arima_diff_2)
```

```
## Series: a10
## ARIMA(0,2,1)
##
## Coefficients:
##      ma1
##      -1.0000
## s.e.    0.0123
##
## sigma^2 estimated as 5.052:  log likelihood=-452.38
## AIC=908.75   AICc=908.81   BIC=915.37
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 0.02151035 2.231053 1.304449 -1.514552 12.8114 1.007087 -0.3650991
```

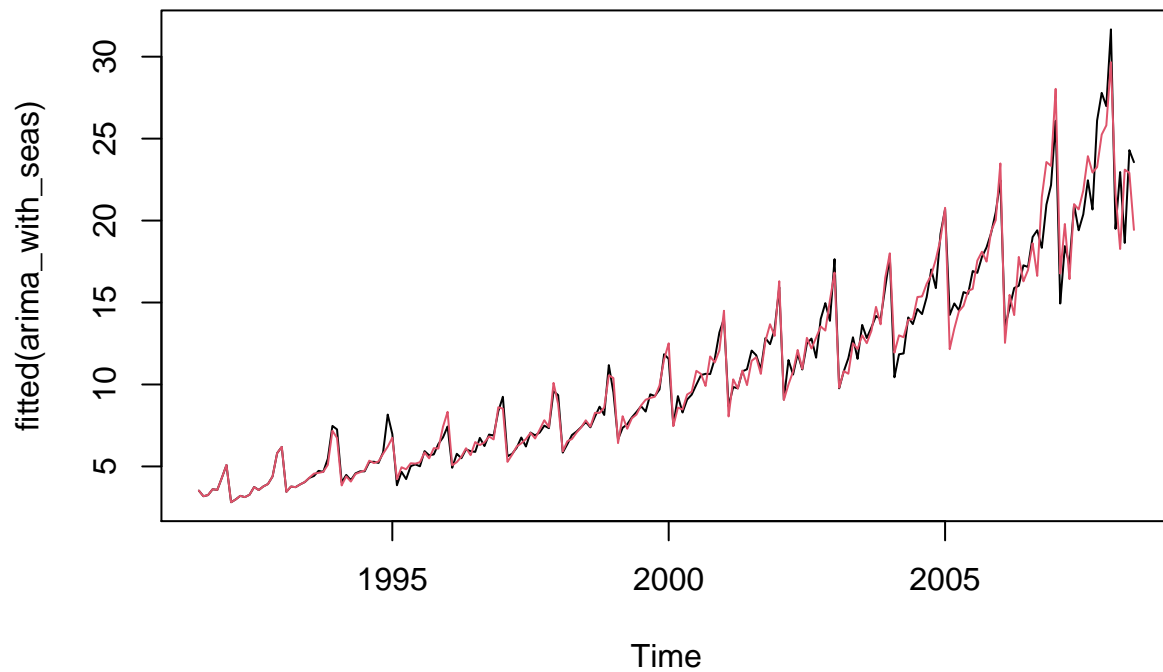
```
acf(residuals(arima_diff_2))
```


Series residuals(arima_diff_2)



#EXERCISE (SECOND TRY)

```
arima_with_seas<- Arima(a10, order=c(0,1,1), seasonal=list(order=c(0,2,1), period=12))  
plot(fitted(arima_with_seas))  
lines(a10, col=2)
```

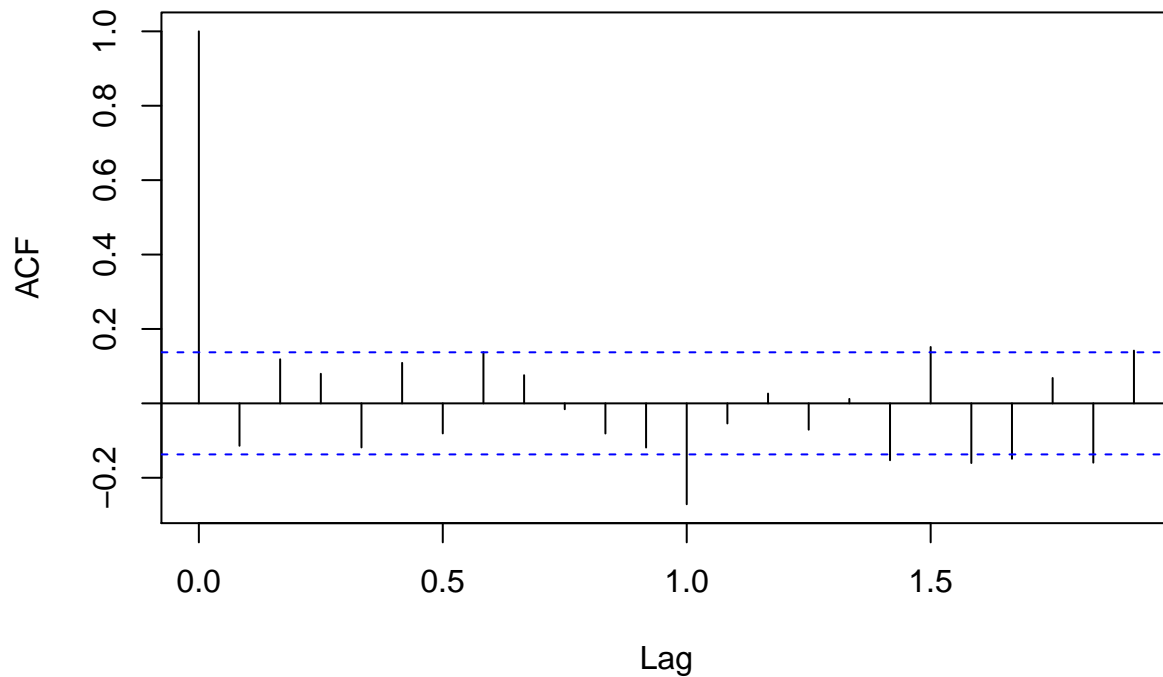


```
summary(arima_with_seas)
```

```
## Series: a10
## ARIMA(0,1,1)(0,2,1)[12]
##
## Coefficients:
##          ma1      sma1
##       -0.7914 -0.9977
## s.e.    0.0552  0.0681
##
## sigma^2 estimated as 1.108: log likelihood=-282.9
## AIC=571.79  AICc=571.93  BIC=581.35
##
## Training set error measures:
##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.006429972 0.9805994 0.5878664 -0.1884376 4.531473 0.4538562
##              ACF1
## Training set -0.1139636
```

```
acf(residuals(arima_with_seas))
```

Series residuals(arima_with_seas)



LINEAR MODEL

```
#Linear model with initial data
```

```
a0 <- 1:204
```

```
m1 <- lm(a10~a0)
```

```
summary(m1)
```

```
##
```

```
## Call:
```

```
## lm(formula = a10 ~ a0)
```

```
##
```

```
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
```

```
## -5.0153 -1.5800 -0.3347  1.1413  9.9670
```

```
##
```

```
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 1.130721    0.319622    3.538 0.000501 ***
```

```
## a0           0.093304    0.002704   34.509 < 2e-16 ***
```

```
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

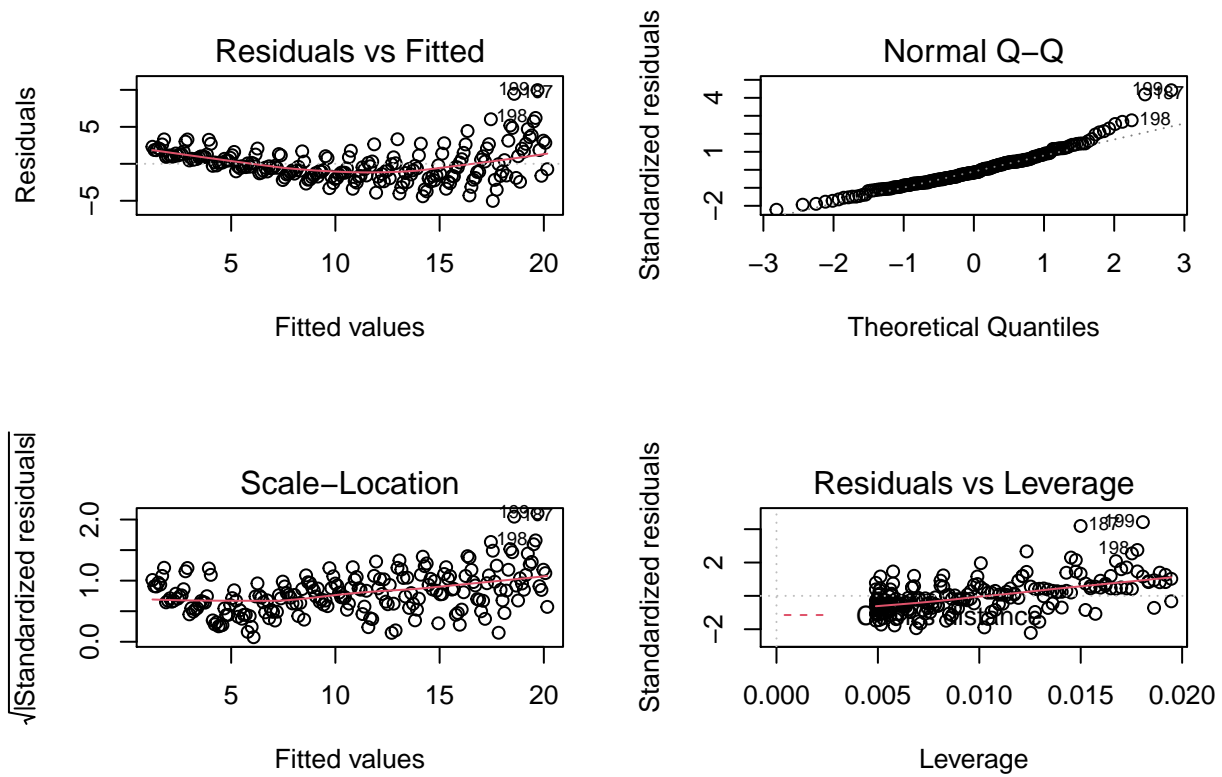
```
##
```

```
## Residual standard error: 2.274 on 202 degrees of freedom
```

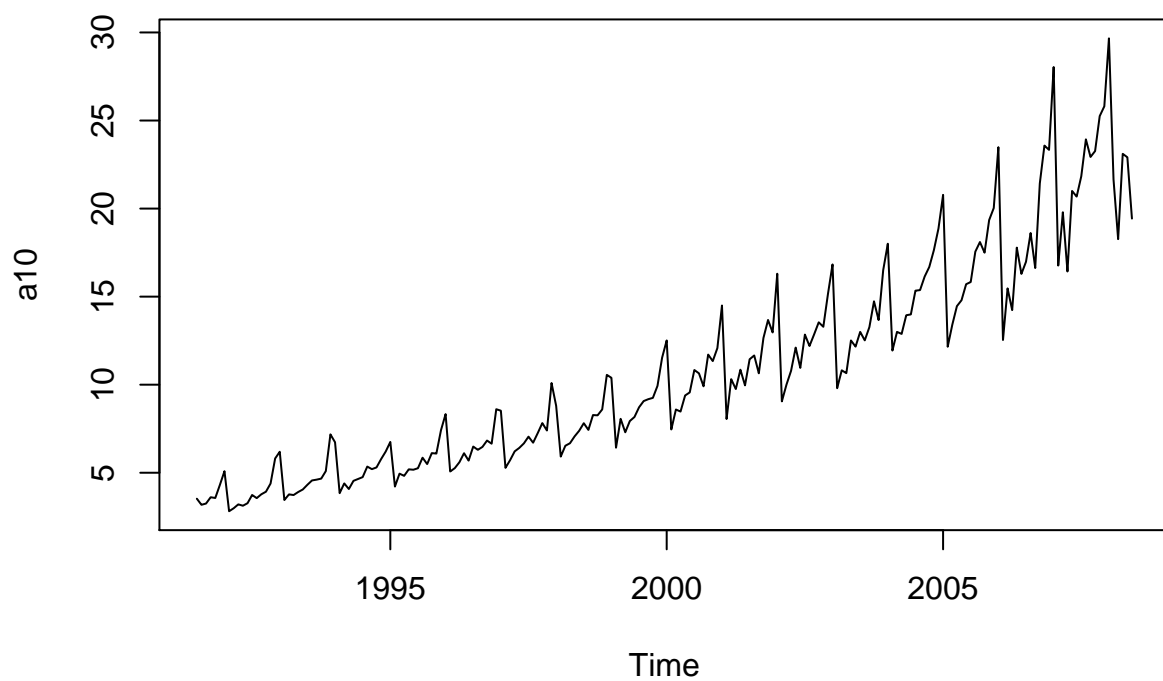
```
## Multiple R-squared:  0.855, Adjusted R-squared:  0.8543
```

```
## F-statistic: 1191 on 1 and 202 DF, p-value: < 2.2e-16
```

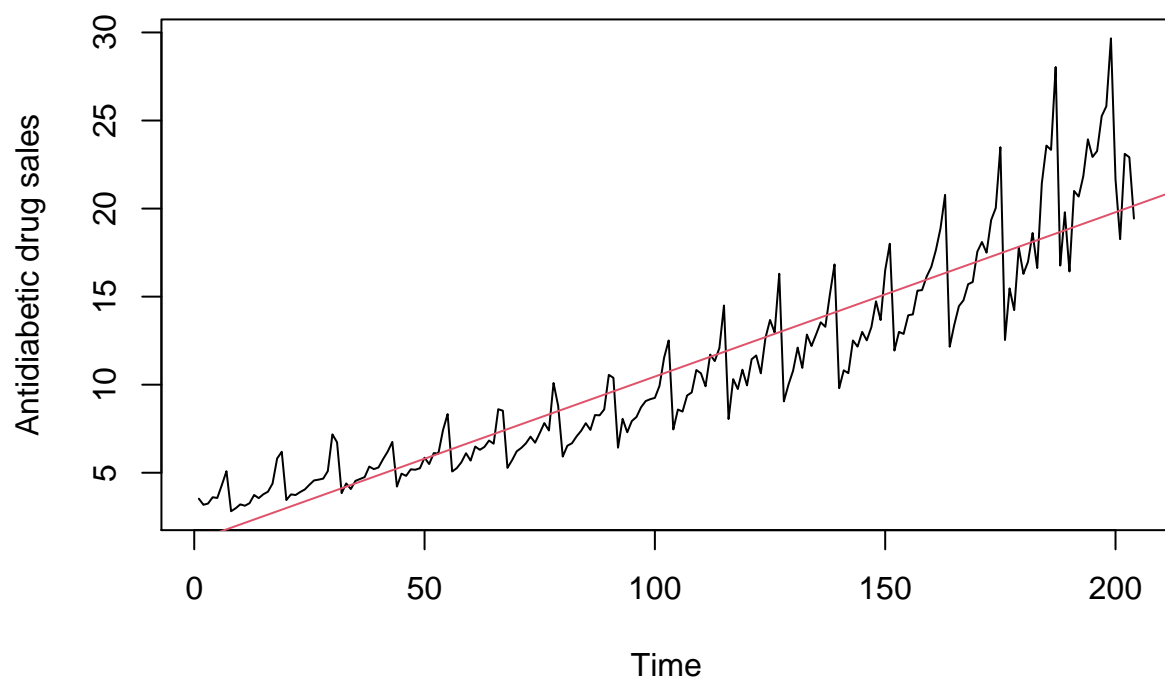
```
#Plot  
par(mfrow=c(2,2))  
plot(m1)
```



```
par(mfrow=c(1,1))  
plot(a10)
```

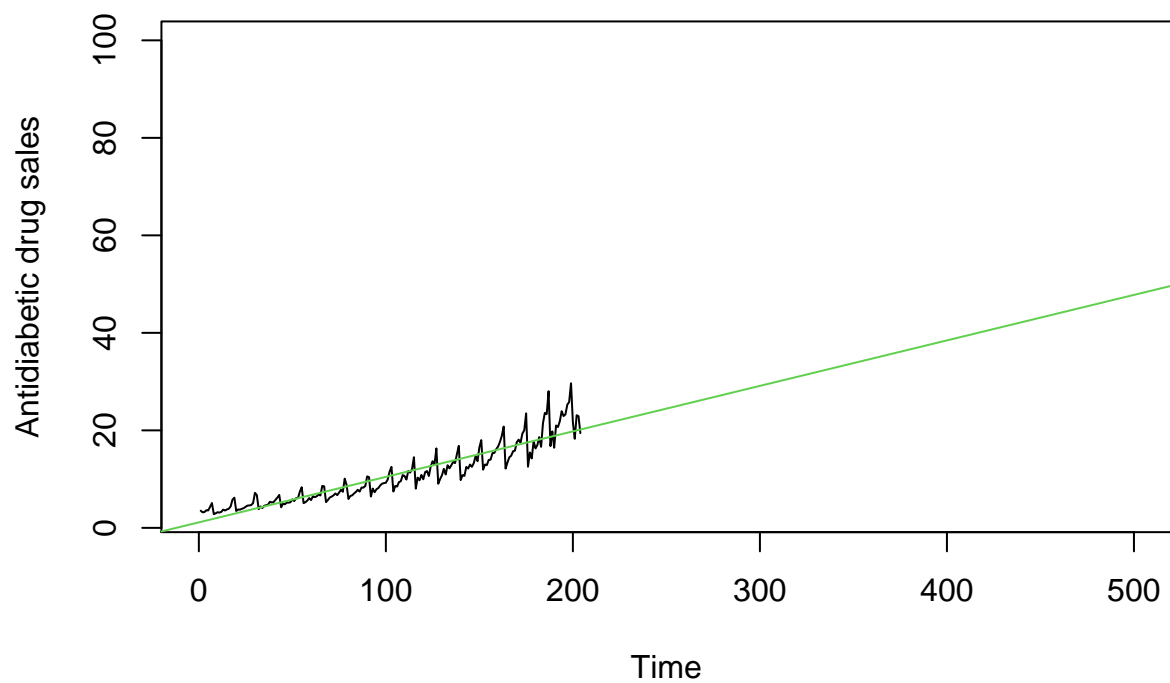


```
plot(as.numeric(a10), type="l", xlab="Time", ylab="Antidiabetic drug sales")  
abline(m1, col=2)
```

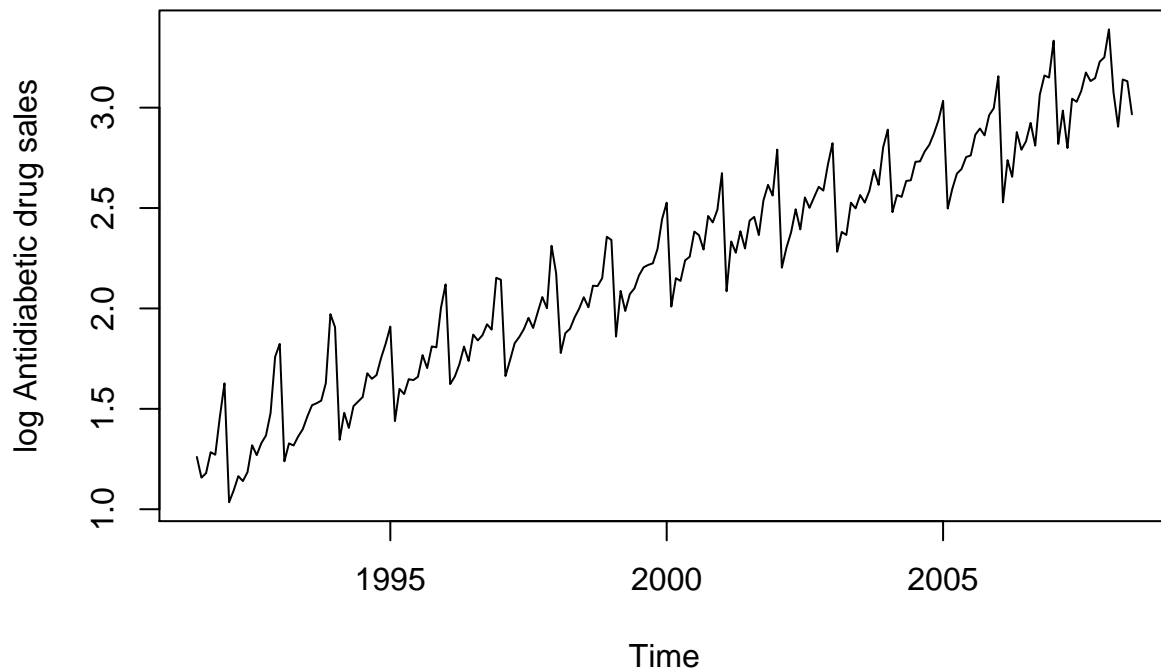


```
#Plot to see the projection
```

```
plot(as.numeric(a10), type="l", xlim=c(0,500), ylim=c(3,100), xlab="Time", ylab="Antidiabetic drug sales",  
abline(m1, col=3)
```



```
#Log transformation  
la10 <- log(a10)  
plot(la10, xlab="Time", ylab="log Antidiabetic drug sales")
```



```
#Linear model with log data
```

```
m2 <- lm(la10~a0)
```

```
summary(m2)
```

```
##
```

```
## Call:
```

```
## lm(formula = la10 ~ a0)
```

```
##
```

```
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.36954 -0.09621 -0.00889  0.07139  0.43395
```

```
##
```

```
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.2577135  0.0216920  57.98  <2e-16 ***
## a0           0.0093211  0.0001835  50.80  <2e-16 ***
```

```
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
```

```
## Residual standard error: 0.1543 on 202 degrees of freedom
```

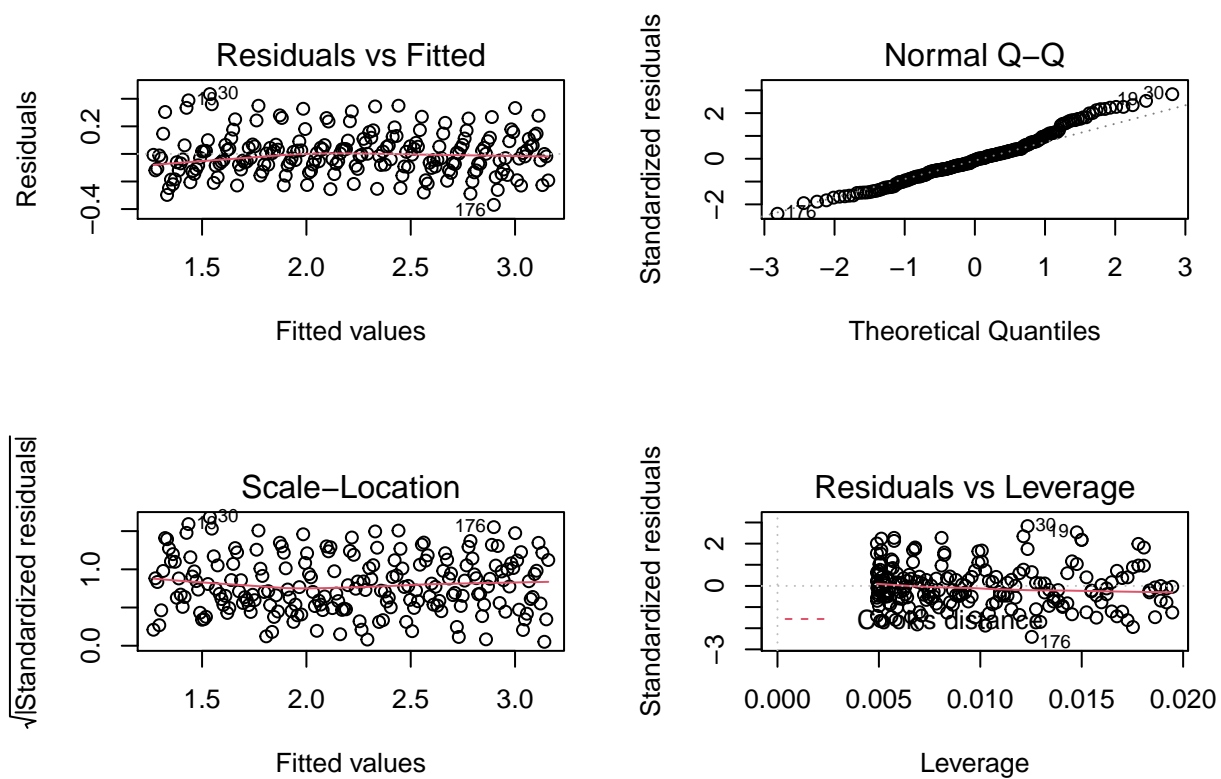
```
## Multiple R-squared:  0.9274, Adjusted R-squared:  0.927
```

```
## F-statistic: 2580 on 1 and 202 DF, p-value: < 2.2e-16
```

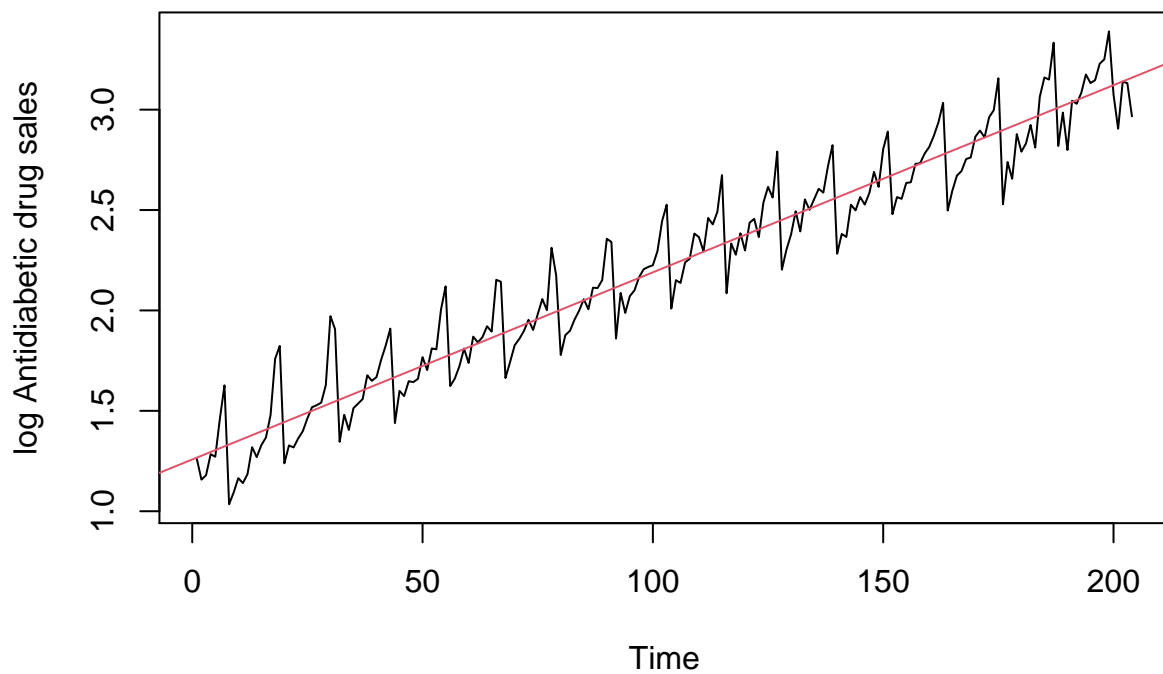
```
#Plot
```

```
par(mfrow=c(2,2))
```

```
plot(m2)
```

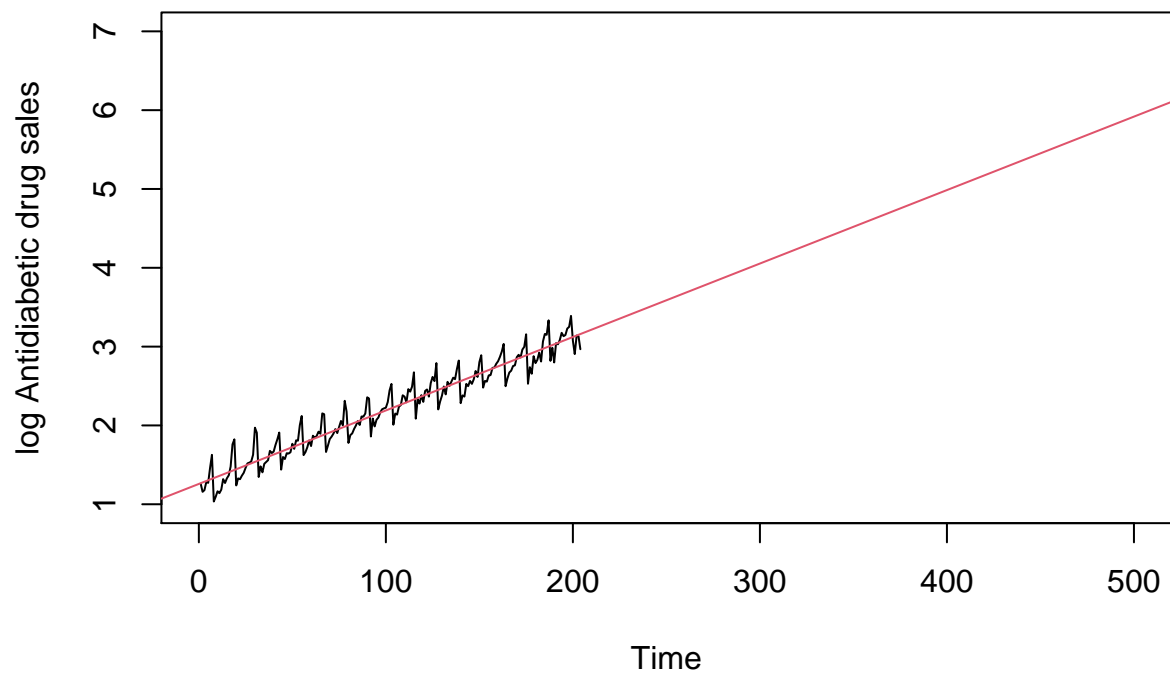



```
par(mfrow=c(1,1))
plot(as.numeric(la10), type="l", xlab="Time", ylab="log Antidiabetic drug sales")
abline(m2, col=2)
```

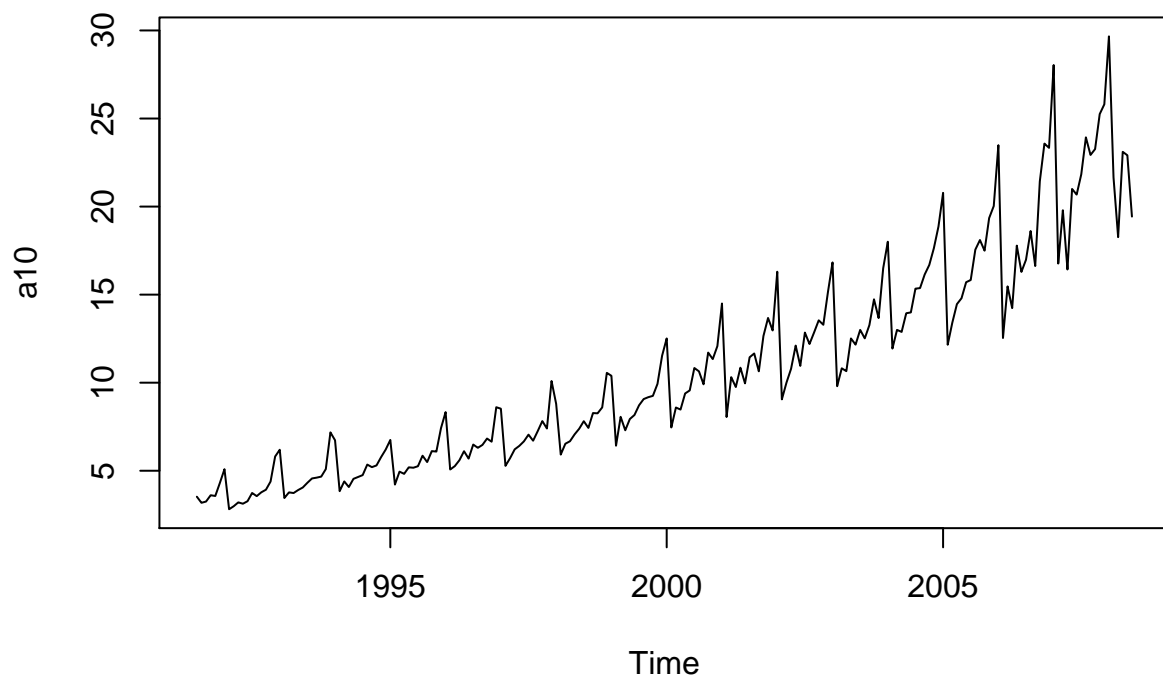


```
#Plot to see the projection
```

```
plot(as.numeric(la10), type="l", xlim=c(0,500), ylim=c(1,7), xlab="Time", ylab="log Antidiabetic drug s  
abline(m2, col=2)
```



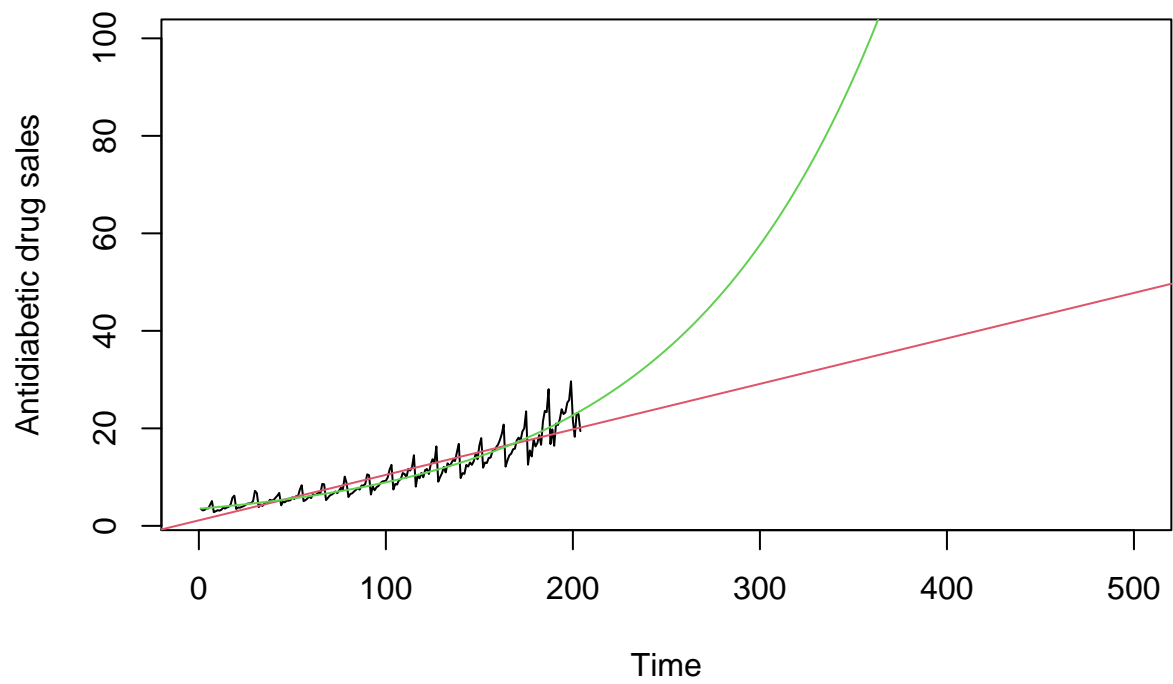
```
#Projections of Linear model and Log Linear model comparison  
plot(a10)
```



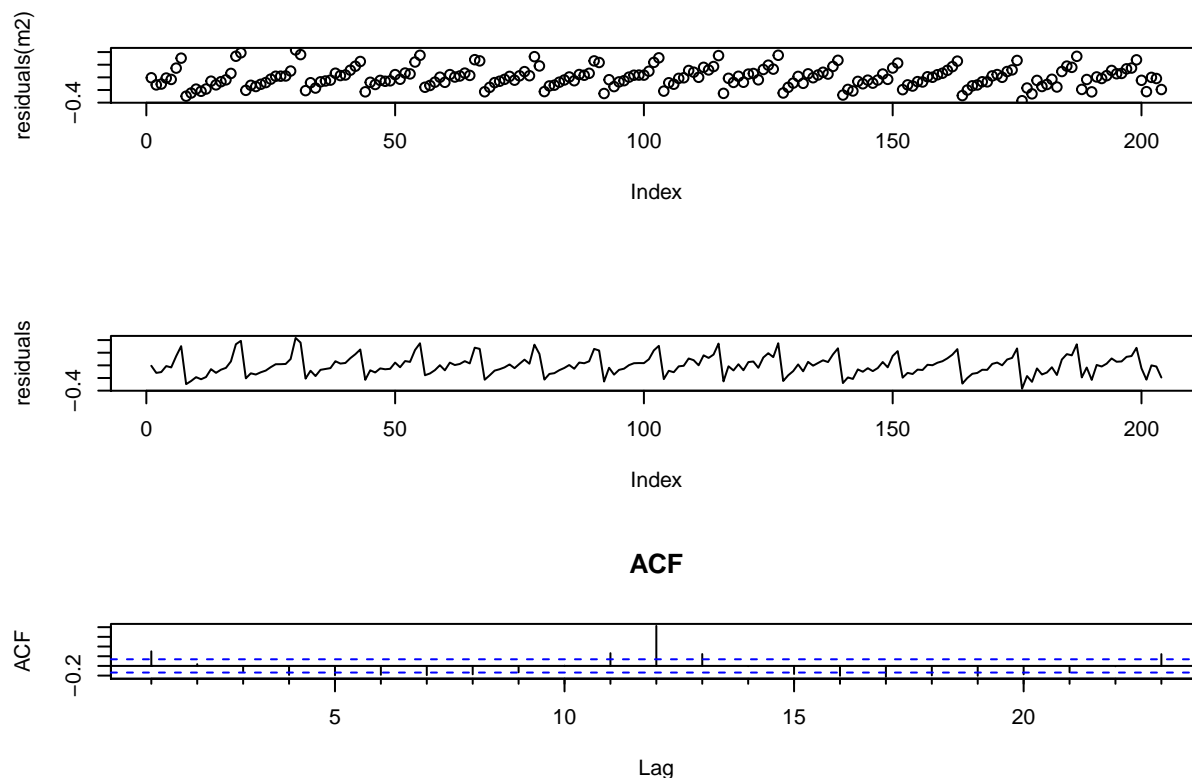
```
plot(as.numeric(a10), type="l", xlim=c(0,500), ylim=c(3,100), xlab="Time", ylab="Antidiabetic drug sales")
abline(m1, col=2)
summary(m2)
```

```
##
## Call:
## lm(formula = la10 ~ a0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.36954 -0.09621 -0.00889  0.07139  0.43395
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.2577135   0.0216920   57.98  <2e-16 ***
## a0           0.0093211   0.0001835   50.80  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1543 on 202 degrees of freedom
## Multiple R-squared:  0.9274, Adjusted R-squared:  0.927
## F-statistic: 2580 on 1 and 202 DF, p-value: < 2.2e-16
```

```
lines(1:500, exp(predict(m2, newdata=data.frame(a0=c(1:500)))), col=3)
```



```
#Exploration of m2 residuals and autocorrelation function  
par(mfrow=c(3,1))  
plot(residuals(m2))  
plot(residuals(m2), type="l", ylab="residuals")  
Acf(residuals(m2), main="ACF")
```



LAB 3

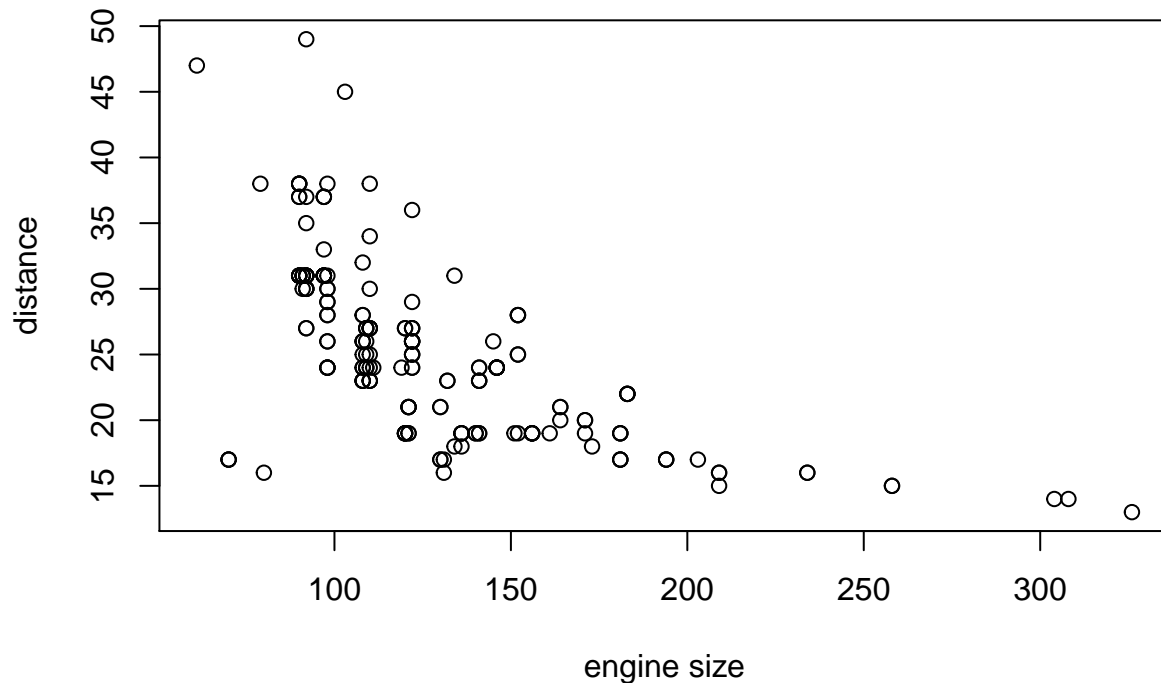
```
auto <- read.csv("auto-us.csv")
str(auto)
```

```
## 'data.frame':    205 obs. of  28 variables:
## $ X              : int  1 2 3 4 5 6 7 8 9 10 ...
## $ symboling      : int  3 3 1 2 2 2 1 1 1 0 ...
## $ normalized.losses: int  NA NA NA 164 164 NA 158 NA 158 NA ...
## $ make           : chr  "alfa-romero" "alfa-romero" "alfa-romero" "audi" ...
## $ fuel.type      : chr  "gas" "gas" "gas" "gas" ...
## $ aspiration      : chr  "std" "std" "std" "std" ...
## $ num.of.doors    : chr  "two" "two" "two" "four" ...
## $ body.style      : chr  "convertible" "convertible" "hatchback" "sedan" ...
## $ drive.wheels    : chr  "rwd" "rwd" "rwd" "fwd" ...
## $ engine.location : chr  "front" "front" "front" "front" ...
## $ wheel.base      : num  88.6 88.6 94.5 99.8 99.4 ...
## $ length          : num  169 169 171 177 177 ...
## $ width           : num  64.1 64.1 65.5 66.2 66.4 66.3 71.4 71.4 71.4 67.9 ...
## $ height          : num  48.8 48.8 52.4 54.3 54.3 53.1 55.7 55.7 55.9 52 ...
## $ curb.weight     : int  2548 2548 2823 2337 2824 2507 2844 2954 3086 3053 ...
## $ engine.type     : chr  "dohc" "dohc" "ohcv" "ohc" ...
## $ num.of.cylinders : chr  "four" "four" "six" "four" ...
```

```
## $ engine.size      : int  130 130 152 109 136 136 136 136 131 131 ...
## $ fuel.system      : chr  "mpfi" "mpfi" "mpfi" "mpfi" ...
## $ bore             : num  3.47 3.47 2.68 3.19 3.19 3.19 3.19 3.19 3.13 3.13 ...
## $ stroke           : num  2.68 2.68 3.47 3.4 3.4 3.4 3.4 3.4 3.4 3.4 ...
## $ compression.ratio: num  9 9 9 10 8 8.5 8.5 8.5 8.3 7 ...
## $ horsepower       : int  111 111 154 102 115 110 110 110 140 160 ...
## $ peak.rpm         : int  5000 5000 5000 5500 5500 5500 5500 5500 5500 5500 ...
## $ city.mpg         : int  21 21 19 24 18 19 19 19 17 16 ...
## $ highway.mpg      : int  27 27 26 30 22 25 25 25 20 22 ...
## $ price            : int  13495 16500 16500 13950 17450 15250 17710 18920 23875 NA ...
## $ N.cylinders      : int  4 4 6 4 5 5 5 5 5 5 ...
```

```
#Analysis of variable engine.size
y <- auto$city.mpg
x <- auto$engine.size

#Preliminary plot
plot(x,y,xlab="engine size", ylab="distance")
```



LOCAL REGRESSION

```
#install.packages("sm")
library(sm)
```

```
## Package 'sm', version 2.2-5.6: type help(sm) for summary information
```

```
plot(x,y)
```

```
#sm.regression: nonparametric regression estimate function (performs local regression)
```

```
#h: smoothing parameter
```

```
#add=T: I can add other lines after the creation of the plot
```

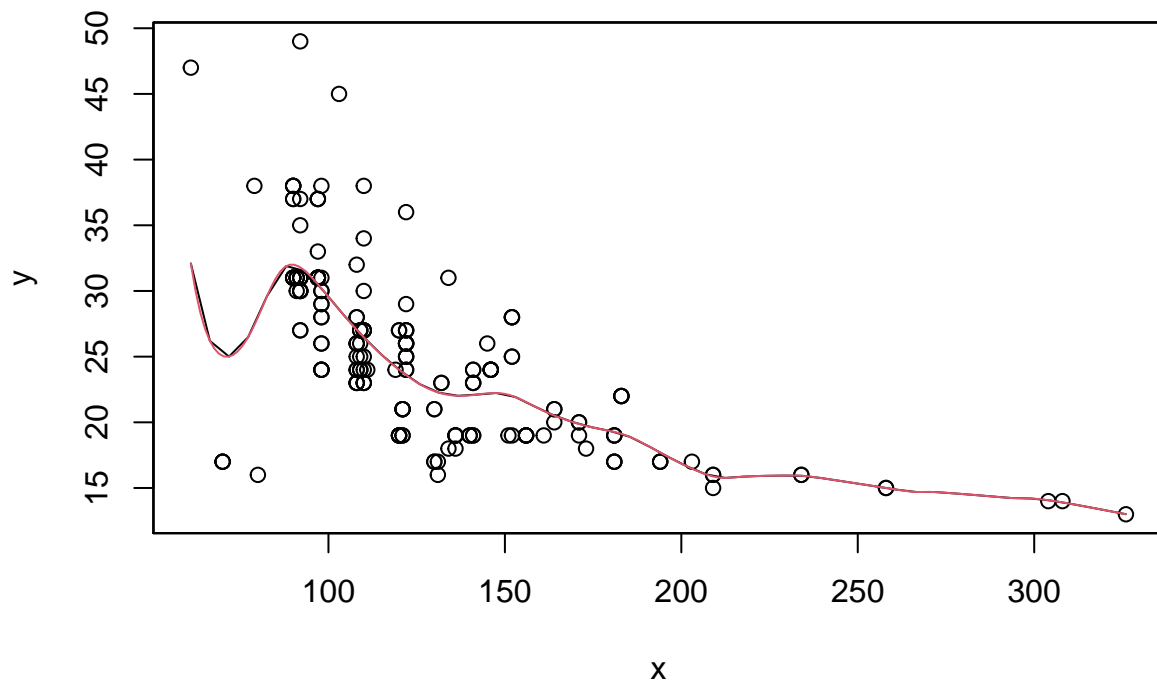
```
#We can see at the beginning of our local regr. line a structure made by straight lines (we have local
```

```
#straight lines), if we want a smoother function we should add the number of points
```

```
sm.regression(x, y, h = 10, add = T)
```

```
#Increase the number of points where the function is estimated
```

```
sm.regression(x, y, h = 10, add = T, ngrid=200, col=2)
```



```
#We try with different values for h
```

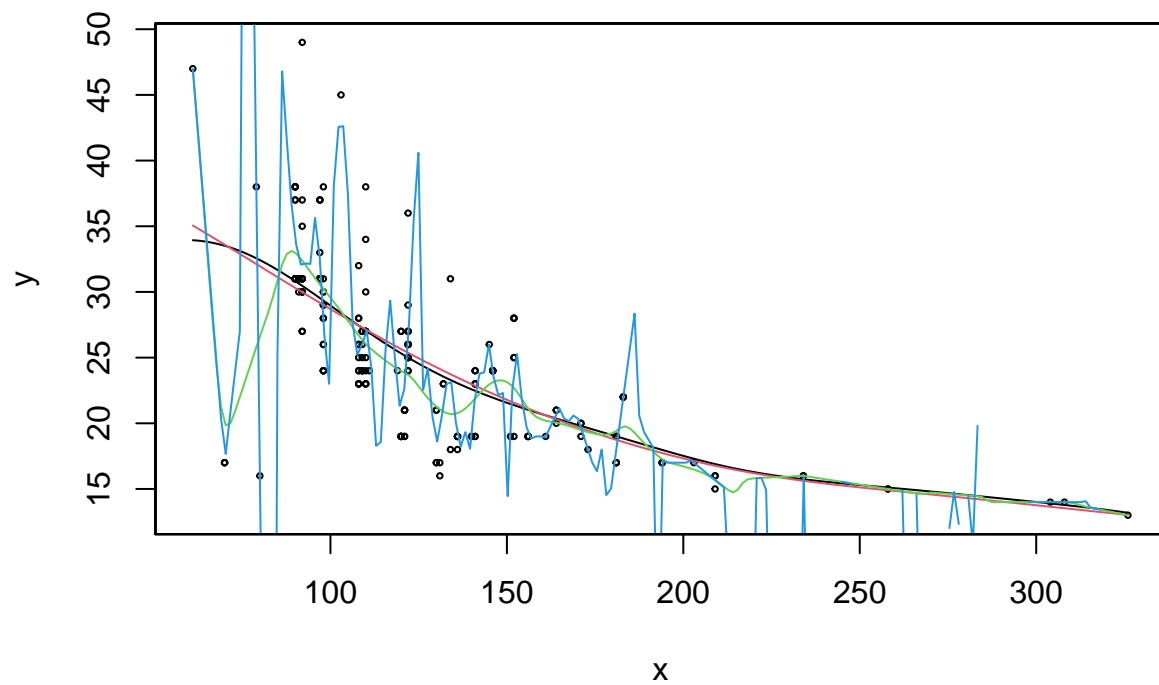
```
#Increasing the value of h we have a smoother function, a lower value of h implies a jumpy function
```

```
sm.regression(x, y, h = 30, ngrid=200, col=1)
```

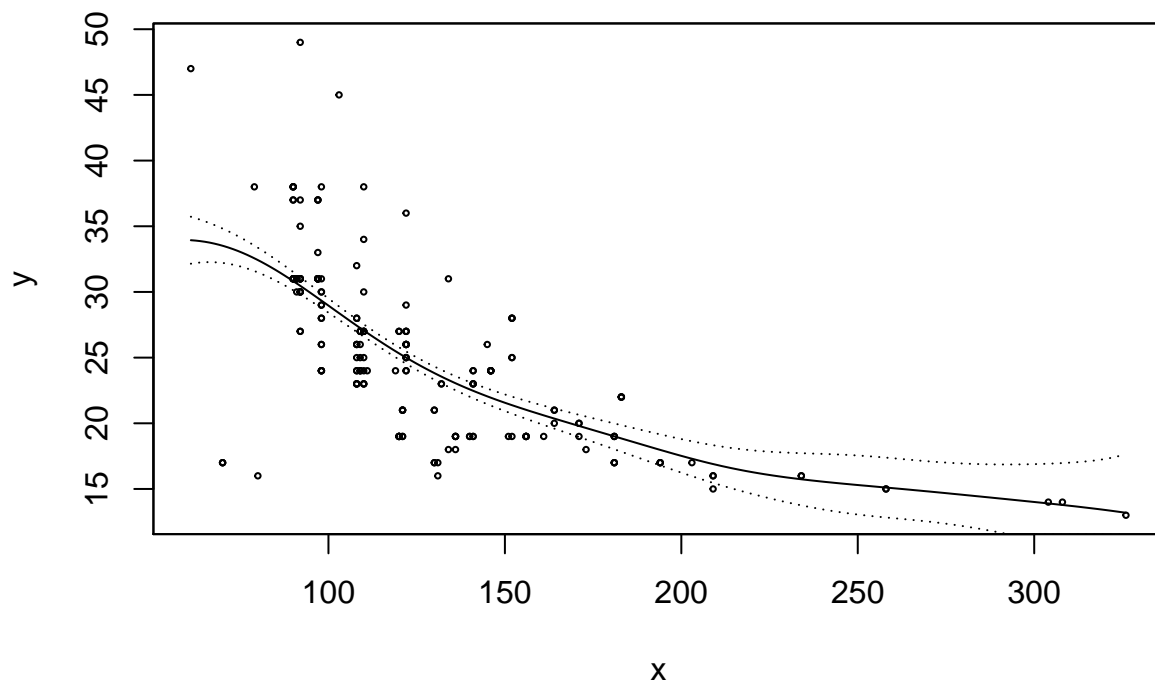
```
sm.regression(x, y, h = 50, add = T, ngrid=200, col=2)
```

```
sm.regression(x, y, h = 5, add = T, ngrid=200, col=3)
```

```
sm.regression(x, y, h = 1, add = T, col=4, ngrid=200)
```

```
#We add variability bands ('se': standar deviation)
sm.regression(x, y, h = 30, ngrid=200, display="se")
```



#The bands are larger when I have few points

LOCAL POLYNOMIAL (Another way to implement local regression)

We will obtain the same results of local regression but it will be performed in a different way in terms of specification that I provide.

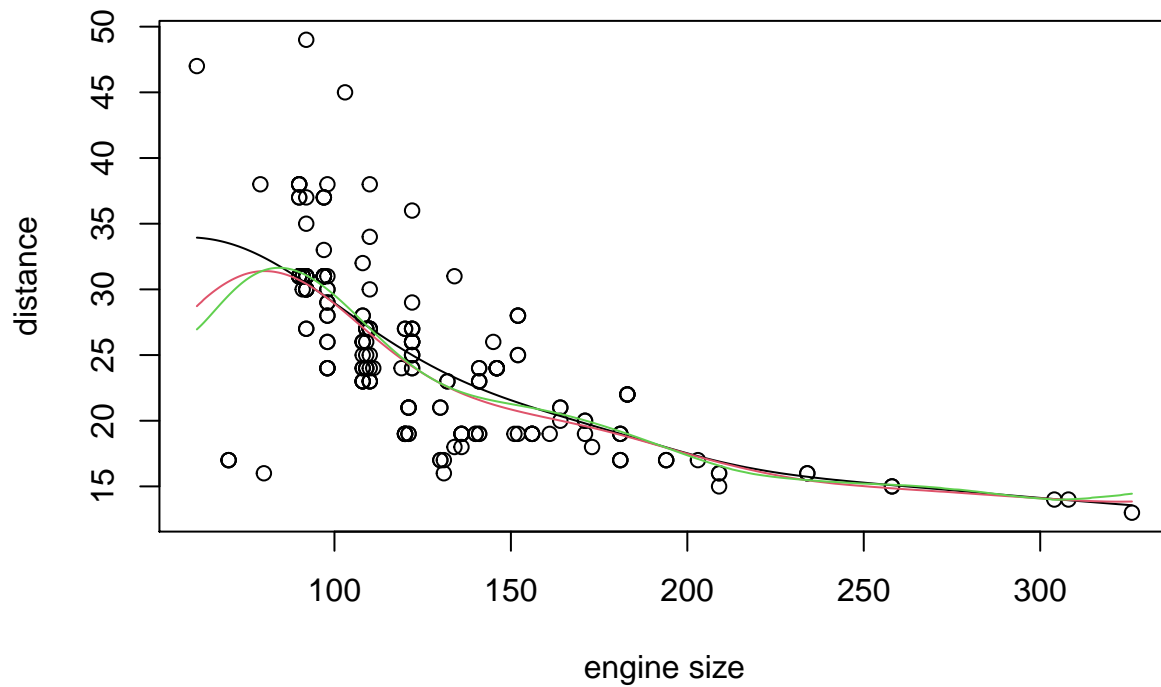
```
#Another library
install.packages("KernSmooth")
library(KernSmooth)
```

```
## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
```

```
plot(x, y, xlab="engine size", ylab="distance")
#plots local regression in an equivalent way (but obtain with different specifications)
a1 <- locpoly(x, y, degree=1, bandwidth=30)
lines(a1)
#We obtain the same result of: "sm.regression(x, y, h = 30, ngrid=200, col=1)"

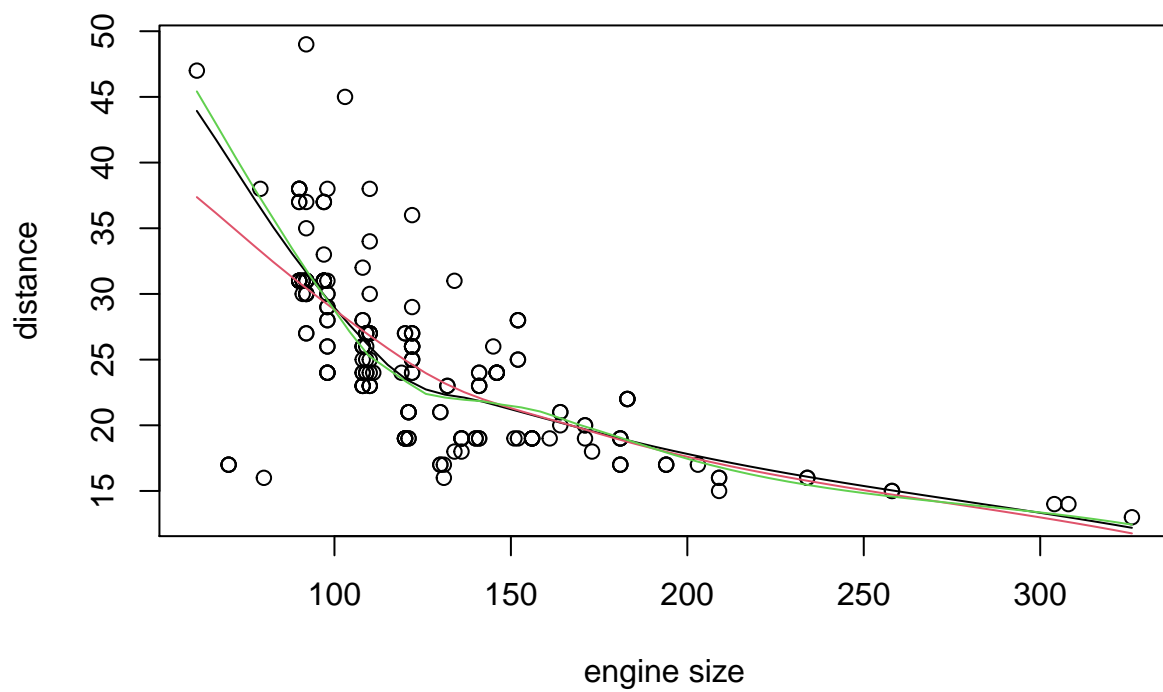
#We may plot different polynomials: d=2 and d=3
a2 <- locpoly(x,y,degree=2,bandwidth=30)
lines(a2,col=2)
```

```
a3 <- locpoly(x,y,degree=3,bandwidth=30)
lines(a3,col=3)
```

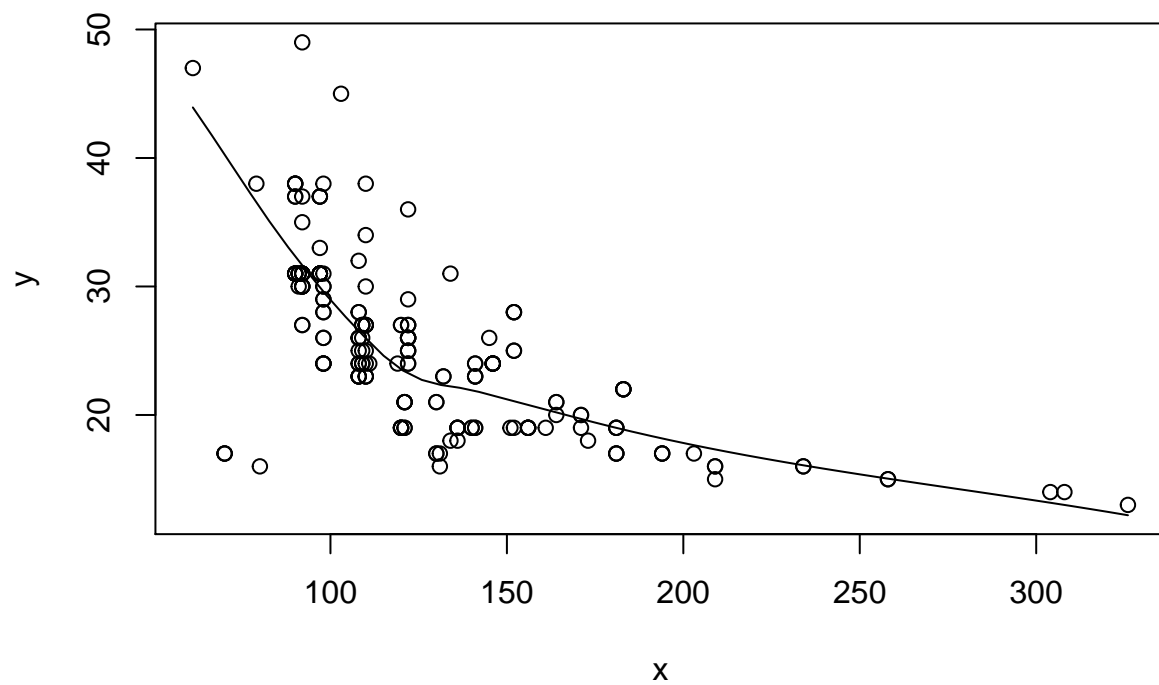


```
# LOESS (no library required, default tool of R)

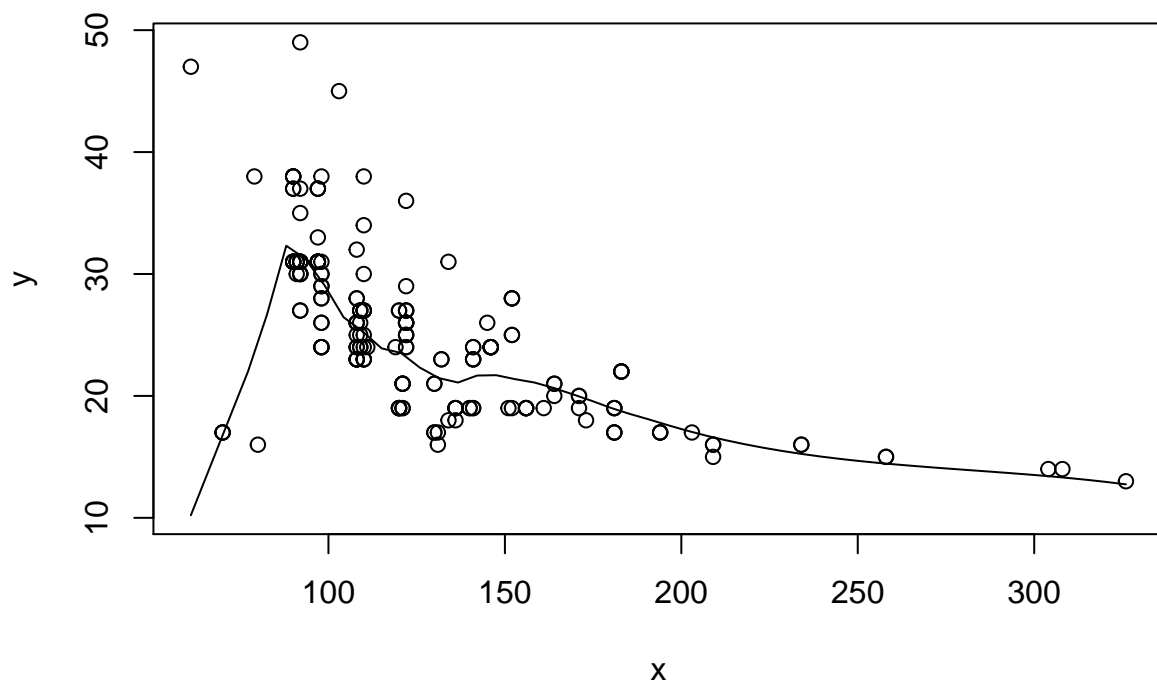
plot(x, y, xlab="engine size", ylab="distance")
#Default span= 2/3
lo1 <- loess.smooth(x,y)
lines(lo1)
#We try with other smoothing parameters 'span'
lo2 <- loess.smooth(x,y,span=0.9)
lines(lo2,col=2)
lo3 <- loess.smooth(x,y,span=0.4)
lines(lo3,col=3)
```



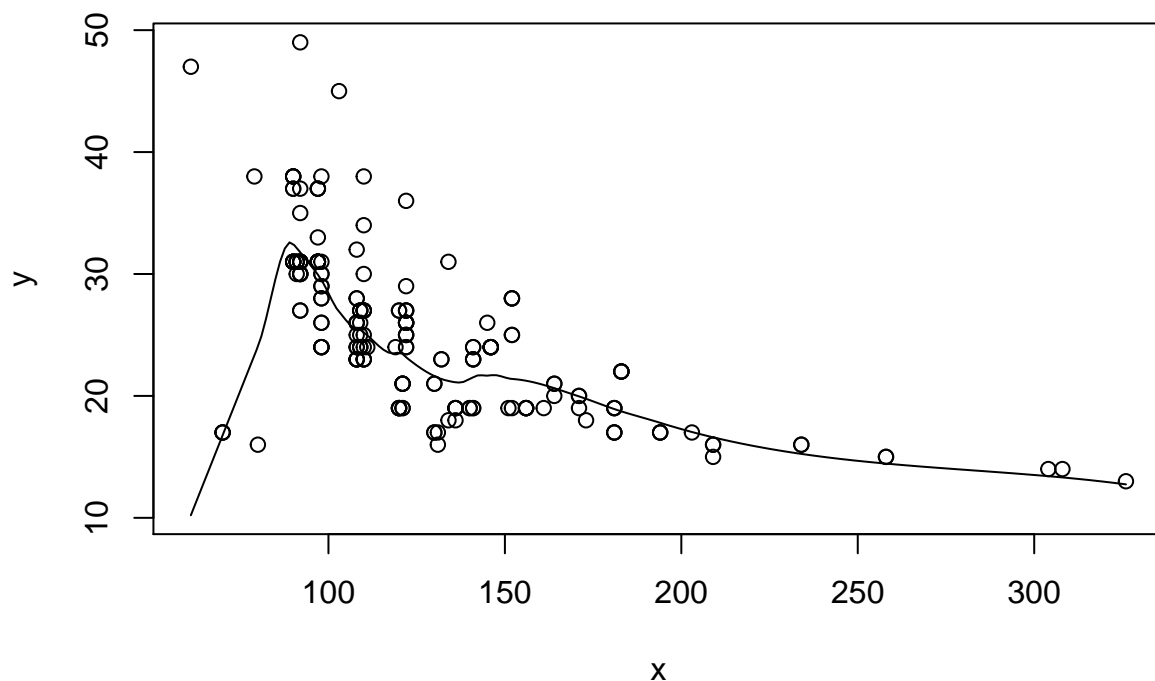
```
#Another way to perform loess; it performs directly the plot
#default span= 2/3, same results of "lo1 <- loess.smooth(x,y)"
scatter.smooth(x,y)
```



```
# same results of "loess.smooth(x,y,span=0.3)"  
scatter.smooth(x,y, span=0.3)
```



```
#adding the 'evaluation' parameter we obtain a smoother line  
scatter.smooth(x,y, span=0.3, evaluation=200)
```



REGRESSION SPLINES (cubic splines)

```
#install.packages("splines")
library(splines)

#We select and identify the knots 'equispaced' (we will consider like "true knots" the internal ones,
#the others are "boundary knots")
xi<-seq(min(x), max(x), length=4)
xi

## [1] 61.0000 149.3333 237.6667 326.0000

#Model (2 internal knots, from 2 to 3. Note: "knots=" define the positions of knots, not the number)
m1<-lm(y ~ bs(x, knots=xi[2:(length(xi)-1)], degree=3))
#For graphical reasons select 200 points where to evaluate the model
xxx<-seq(min(x),max(x),length=200)
#Make predictions by using the 'xxx' points
fit1<-predict(m1, data.frame(x=xxx))

#Plot
plot(x,y,xlab="engine size", ylab="distance")
lines(xxx,fit1,col=2)
#Vertical plots to indicate the knots
```

```

abline(v=xi[2], lty=3)
abline(v=xi[3], lty=3)

#In the same function "bs()", I may select the knots by using the degrees of freedom and the degree
#Basic functions b-spline for a cubic spline (degree=3)
#df directly related to the number of knots
#df=length(knots) + degree, so to select 2 knots we need df=5 and degrees=3;
#The knots are selected by using the quantiles of 'x' distribution

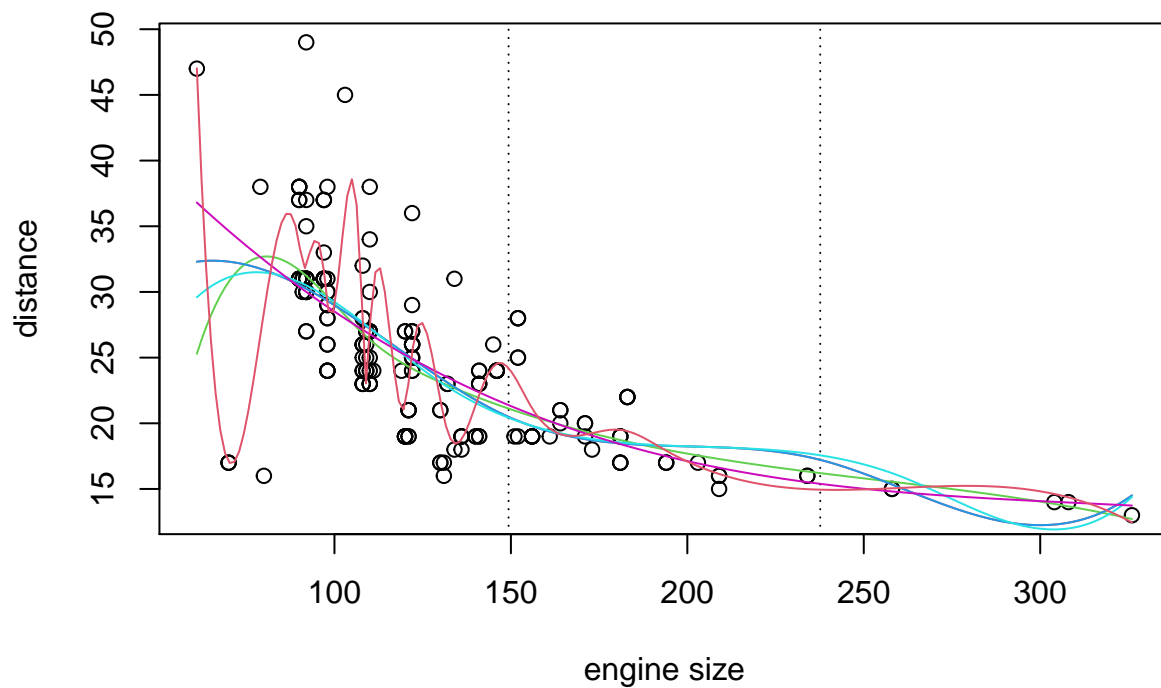
#First model with 2 internal knots (5 (degrees of freedom) - 3 (degrees) = 2 knots)
m1bis<-lm(y~bs(x, df=5, degree=3))
fit1<-predict(m1bis, data.frame(x=xxx))
lines(xxx,fit1,col=3)

#OBSERVATION
#Why do not we have the same results for m1 and m1bis? This is due to the choice of knot positions.
#m1bis considers always 2 knots
#mx: m1 model with df, knots and degree specification (mx overlaps m1)
mx<-lm(y ~ bs(x, df=5, knots=xi[2:(length(xi)-1)], degree=3))
fitx<-predict(mx, data.frame(x=xxx))
#Plot
lines(xxx,fitx,col=4)
#mx2:m1 model with df and degree specification but with 2 other internal knots (mx2 doesn't overlap m1)
mx2<-lm(y ~ bs(x, df=5, knots=c(130,260), degree=3))
fitx2<-predict(mx2, data.frame(x=xxx))
#Plot
lines(xxx,fitx2,col=5)

#Second model with no internal knots (3(df) - 3(degrees) = 0 knots)
m2 <- lm(y ~ bs(x, df=3, degree=3))
fit2<-predict(m2,data.frame(x=xxx))
lines(xxx,fit2,col=6)

#Third model with 17 knots (without specification of the knot positions)
m3<-lm(y~bs(x,df=20,degree=3))
fit3<-predict(m3,data.frame(x=xxx))
lines(xxx,fit3,col=2)

```

```
### SMOOTHING SPLINES (no library required, default tool)

plot(x,y,xlab="engine size", ylab="distance")
#Default lambda = NULL
s <- smooth.spline(x,y)
#We can plot the model to see how it describes the data: lines(s)
#Forecasting
p<- predict(s, x=xxx)
lines(p, col=1)

#We will consider different models with different values for lambda
#Model 1 (overlaps the previous model)
s1 <- smooth.spline(x,y, lambda=0.0001)
#We can plot the model to see how it describes the data: lines(s1, col=2)
#Forecasting of Model 1
p1<- predict(s1, x=xxx)
lines(p1, col=2)

#Model 2
s2 <- smooth.spline(x,y, lambda=0.00001)
#Forecasting
p2<- predict(s2, x=xxx)
lines(p2, col=3)

#Model 3
s3 <- smooth.spline(x,y, lambda=0.01)
```

```

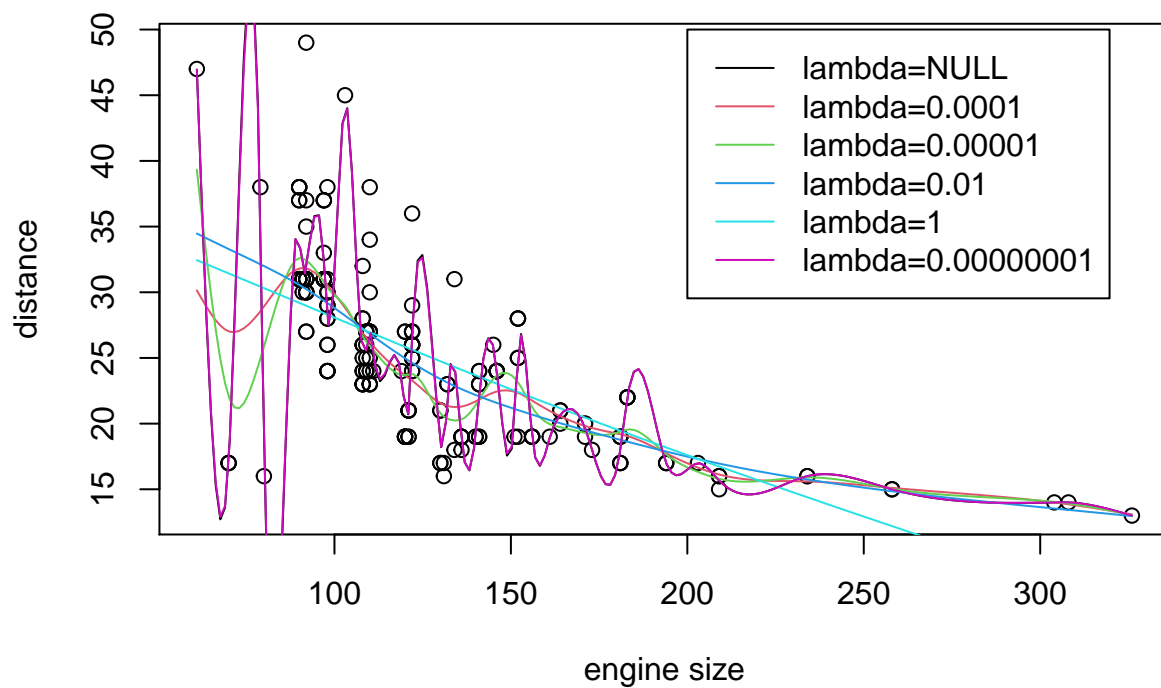
#Forecasting
p3<- predict(s3, x=xxx)
lines(p3, col=4)

#Model 4 (lambda=1 means straight line)
s4 <- smooth.spline(x,y, lambda=1)
#Forecasting
p4<- predict(s4, x=xxx)
lines(p4, col=5)

# Model 5 (the model doesn't allow a lambda value equals to 0)
s5 <- smooth.spline(x,y, lambda=0.00000001)
#Forecasting
p5<- predict(s5, x=xxx)
lines(p5, col=6)

legend(200,50,legend= c("lambda=NULL","lambda=0.0001","lambda=0.00001","lambda=0.01","lambda=1","lambda=0.00000001"))

```



```

#We try with other smoothing parameters like "spar" (alternative to lambda)
plot(x,y,xlab="engine size", ylab="distance")
ss1 <- smooth.spline(x,y,spar=0.8)
pp1<- predict(ss1, x=xxx)
lines(pp1, col=2)

```

