

ARIMA - Ristorante1

- Setting & Function
- Load data and pre-processing
- Data exploration
- Componenti della serie storica
 - Stazionarietà della serie storica
 - Test
 - Variabili stagionali
- Stima modelli
 - Partizionamento dei datasets
 - Modello 1
 - Diagnostica:
 - Previsioni
 - Modello 2
 - Previsioni
 - Modello 3
 - Modello 4
- Confronto modelli
- Evaluation: Confronto forecasting performance dei modelli
 - Scelta misura di *forecasting accuracy*
 - Procedura di *Cross-Validation* con *rolling forecasting origin*
 - Confronto tra i modelli

```
# Clean Workspace
rm(list=ls())
```

Setting & Function

Load data and pre-processing

```
df<- read.csv("../Dati ristoranti//pre-covid_r1.csv", header = TRUE)
```

```
head(df)
```

X...	X id_ristorante	Location	Regione	Provincia	data	scontrini
	<int><int><chr>	<chr>	<chr>	<chr>	<chr>	<dbl>
1	1 245 R000	Montebello	Lombardia	Pavia	2018-09-03	657
2	2 246 R000	Montebello	Lombardia	Pavia	2018-09-04	647
3	3 247 R000	Montebello	Lombardia	Pavia	2018-09-05	635
4	4 248 R000	Montebello	Lombardia	Pavia	2018-09-06	652
5	5 249 R000	Montebello	Lombardia	Pavia	2018-09-07	886
6	6 250 R000	Montebello	Lombardia	Pavia	2018-09-08	1097

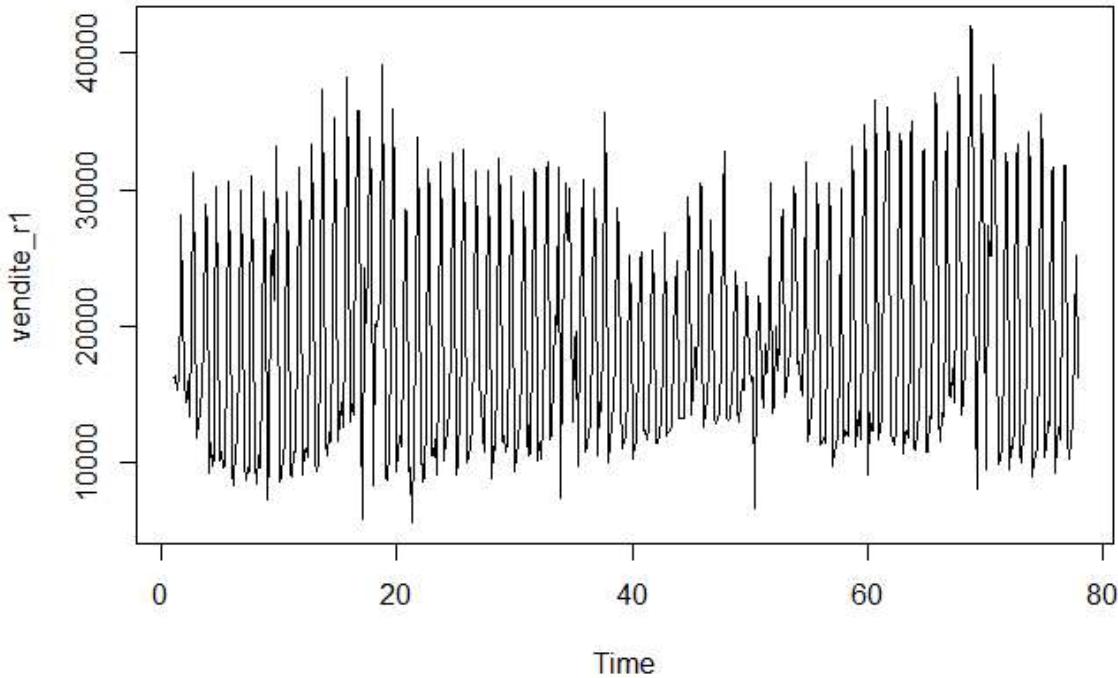
6 rows | 1-9 of 49 columns

```
# Trasformo in serie storica
vendite_r1 <- df[,9]
vendite_r1 <- msts(vendite_r1, seasonal.periods=c(7,365.25), ts.frequency = 7)
# Uso la funzione msts per tenere conto di stagionalità annuale e stagionalità settimanale,
# impostando però la frequenza della serie storica a 7
head(vendite_r1)
```

```
## Multi-Seasonal Time Series:
## Start: 1 1
## Seasonal Periods: 7 365.25
## Data:
## [1] 16201.27 16409.71 15374.10 16531.07 22057.70 28190.46
```

Data exploration

```
plot(vendite_r1)
```



I dati a disposizione sono incassi giornalieri per il Ristorante1. Per dati giornalieri ci possono essere più tipologie di stagionalità. Nel nostro caso abbiamo, molto probabilmente:

- Stagionalità settimanale
- Stagionalità annuale

Per quanto riguarda la stagionalità settimanale, grazie alle variabili a disposizione, possiamo subito verificare se effettivamente è presente nei nostri dati.

```
# Media dei giorni feriali

feriali <- df[df$Weekend == 'False' | df$Festivo == FALSE ,c( "lordototale", "Giorno")]
feriali <- tapply(feriali$lordototale, feriali$Giorno, mean, simplify = FALSE)

# Media dei giorni "weekend"
weekend <- df[df$Weekend == 'True',c("lordototale", "Giorno")]
weekend <- tapply(weekend$lordototale, weekend$Giorno, mean, simplify = FALSE)

data.frame(c(feriali,weekend))
```

Monday	Thursday	Tuesday	Wednesday	Friday	Saturday	Sunday
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
12146.36	13991.77	12058.68	13439.79	19922.83	31510.69	27733.93

1 row

Notiamo infatti come le medie giornalieri siano sostanzialmente simili dal Lunedì al Giovedì, mentre aumentano notevolmente nei giorni Venerdì-Sabato-Domenica.

Per valutare la presenza di stagionalità annuale, sfruttiamo la variabile "Festivo", valorizzata a True quando siamo in presenza di un Festivo, che non sia però un giorno del weekend.

```
festivo_noweekend <- df[df$Weekend == 'False',c("data", "lordototale", "Giorno",
                                                 "Festivo")]
tapply(festivo_noweekend$lordototale, festivo_noweekend$Festivo, mean)
```

```
##     False      True
## 12639.92 19018.46
```

Le medie nei giorni festivi risultano essere maggiori rispetto a quelle dei giorni feriali. Ciò quindi ci porta ad affermare la presenza di stagionalità annuale, anche se difficilmente stimabile per via della quantità ridotta di osservazioni.

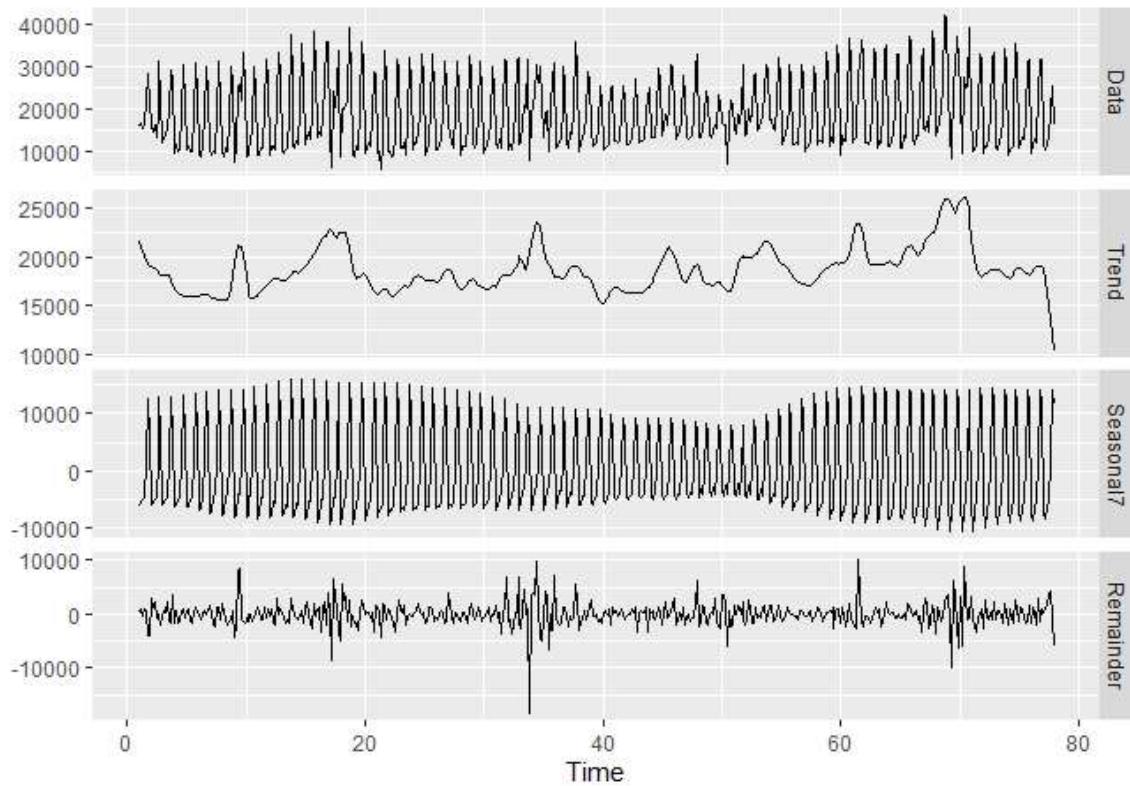
Componenti della serie storica

Andiamo a decomporre la serie storica nelle sue componenti, ovvero: Trend, Stagionalità, eventuale Ciclo

```
# Usiamo la funzione MSL che ammette eventuali stagionalità multiple
vendite_r1_dec <- mstl(vendite_r1)
```

```
## Warning in mstl(vendite_r1): Dropping seasonal components with fewer than two
## full periods.
```

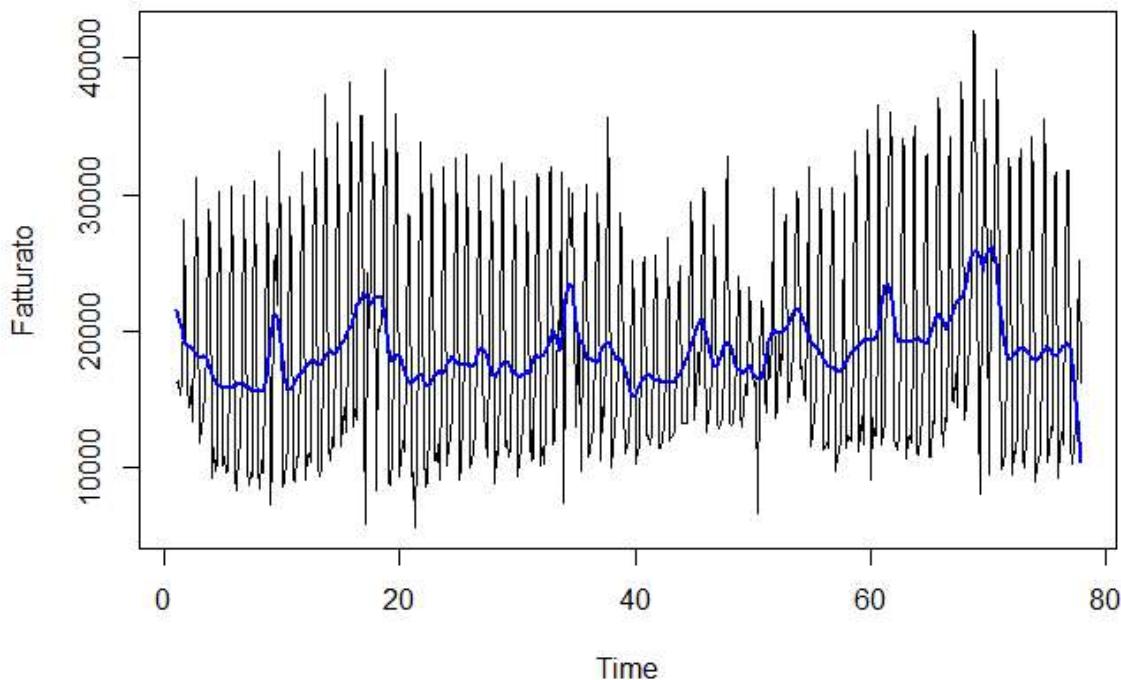
```
autoplot(vendite_r1_dec)
```



Come precedentemente accennato la stagionalità a 365 giorni non viene stimata per insufficienza di dati (servono almeno due periodi completi). Notiamo la presenza di un trend irregolare, in quanto include la stagionalità annuale non stimata, ma che comunque sembra crescente se non nell'ultima parte della serie storica, dove probabilmente i dati sono già influenzati dall'avvento della pandemia. La stagionalità settimanale ci indica come nella parte centrale della serie storica i picchi settimanali sono stati meno marcati rispetto al resto della serie storica.

```
plot(vendite_r1, main="Trend Vendite giornaliere", xlab ="Time", ylab="Fatturato")
lines(vendite_r1_dec[,2], col="blue", lwd=2)
```

Trend Vendite giornaliere



Notiamo come effettivamente le irregolarità del trend sono dovute a tutte le festività non modellate dalla stagionalità settimanale

Stazionarietà della serie storica

La serie storica è ovviamente non stazionaria, per i seguenti motivi:

- Presenza di stagionalità (non stazionarietà stagionale)
- Presenza di un trend (osservazioni sistematicamente sopra o sotto la media)
- Possibile non stazionarietà in varianza (se guardiamo il grafico della decomposizione, la parte stagionale tende ad un certo punto a restringersi)

```
ndiffs(vendite_r1)
```

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

```
nsdiffs(vendite_r1)
```

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

```
nsdiffs(diff(vendite_r1))
```

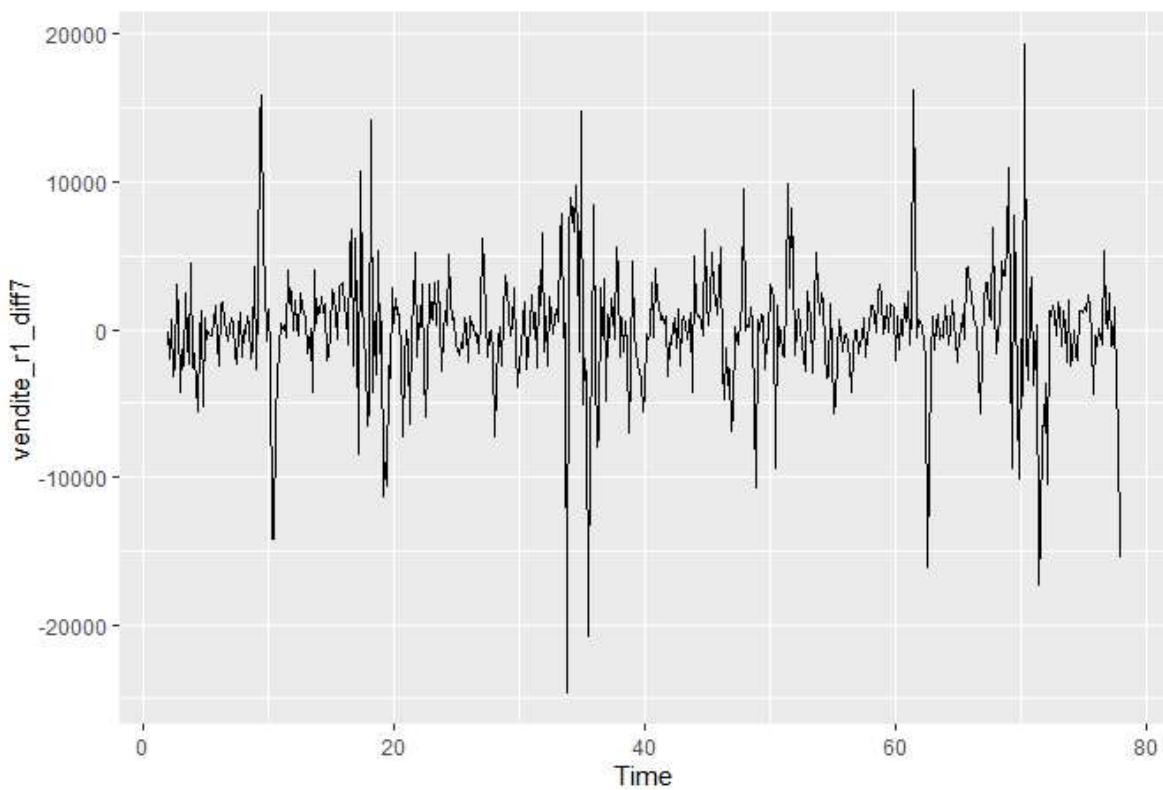
```
## [1] 1
```

```
kpss.test(vendite_r1)
```

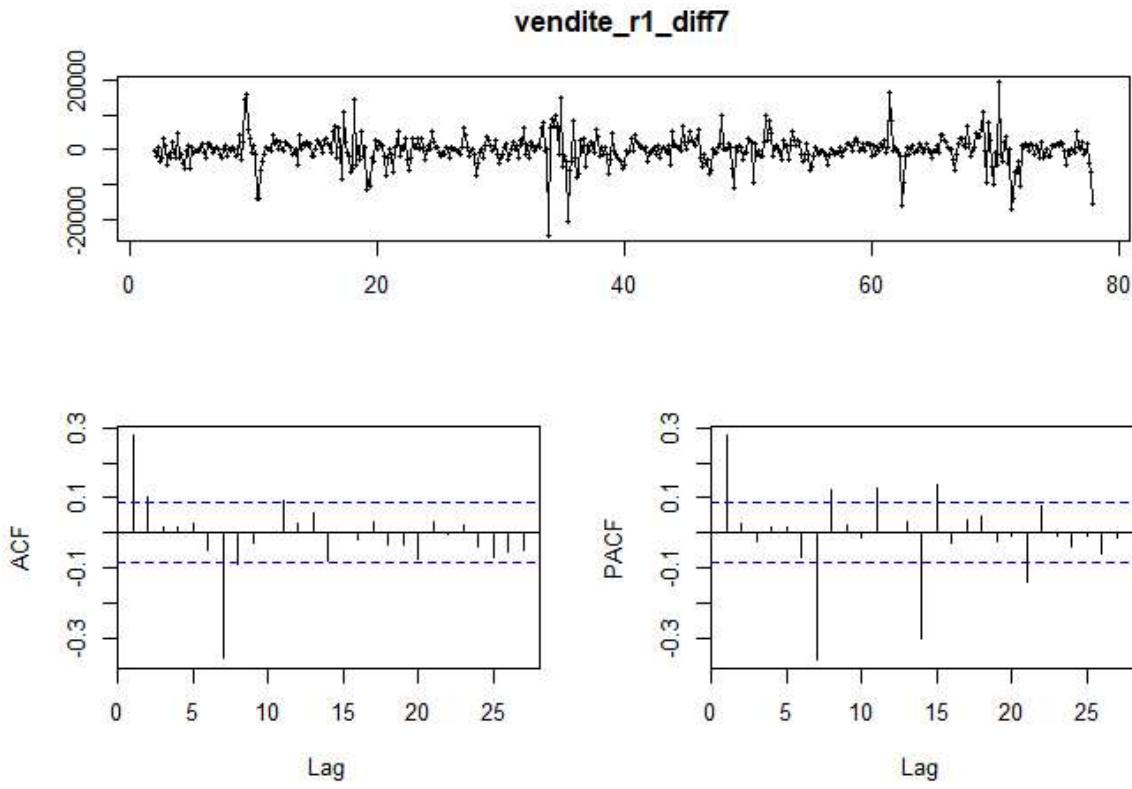
```
## Warning in kpss.test(vendite_r1): p-value smaller than printed p-value
```

```
##  
## KPSS Test for Level Stationarity  
##  
## data: vendite_r1  
## KPSS Level = 1.2529, Truncation lag parameter = 6, p-value = 0.01
```

```
vendite_r1_diff7 <- diff(vendite_r1,7)  
autoplot(vendite_r1_diff7)
```



```
tsdisplay(vendite_r1_diff7)
```



```
ndiffs(vendite_r1_diff7)
```

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

Non sono ora necessarie altre differenze.

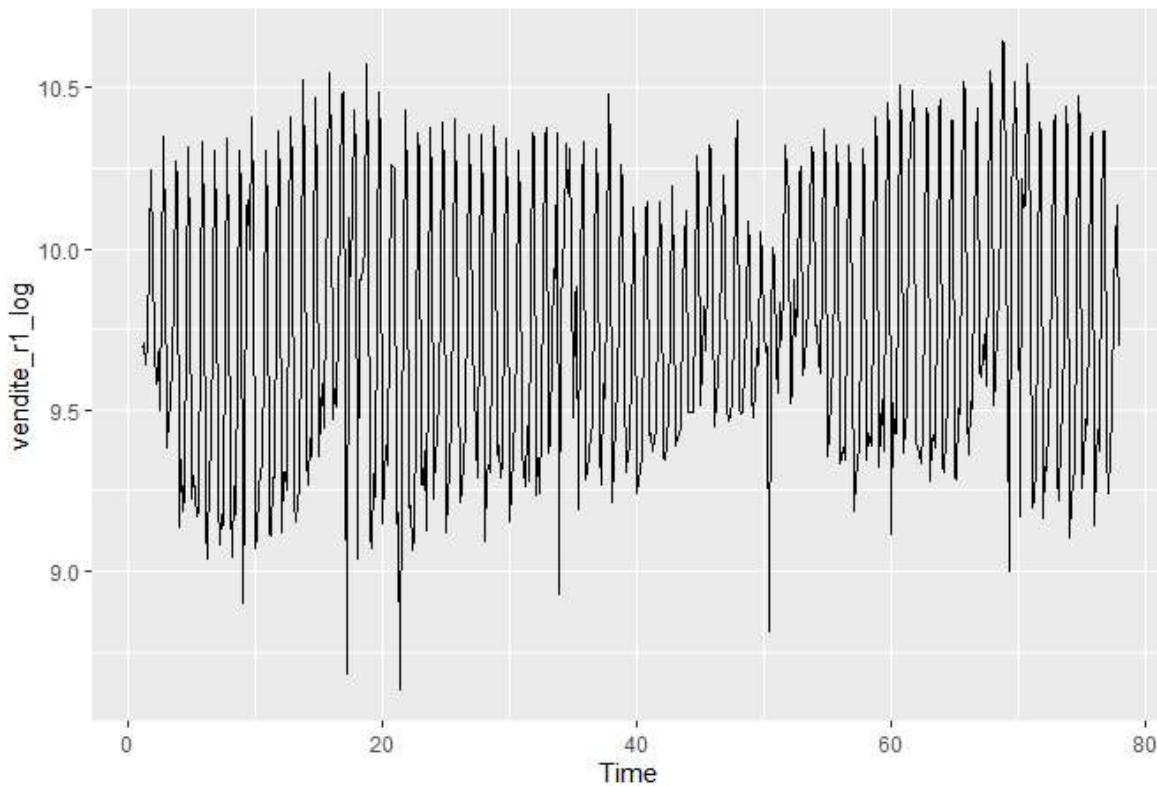
Andiamo a valutare se la non stazionarietà è risolvibile attraverso una trasformazione di BoxCox dei dati

```
BoxCox.lambda(vendite_r1)
```

```
## [1] 0.4671564
```

La stima del lambda ottimale è vicina allo zero (trasformazione log). Valutiamo quindi come cambiano le considerazioni di prima applicando una trasformazione logaritmica ai dati

```
vendite_r1_log <- log(vendite_r1)
autoplot(vendite_r1_log)
```



```
ndiffs(vendite_r1_log)
```

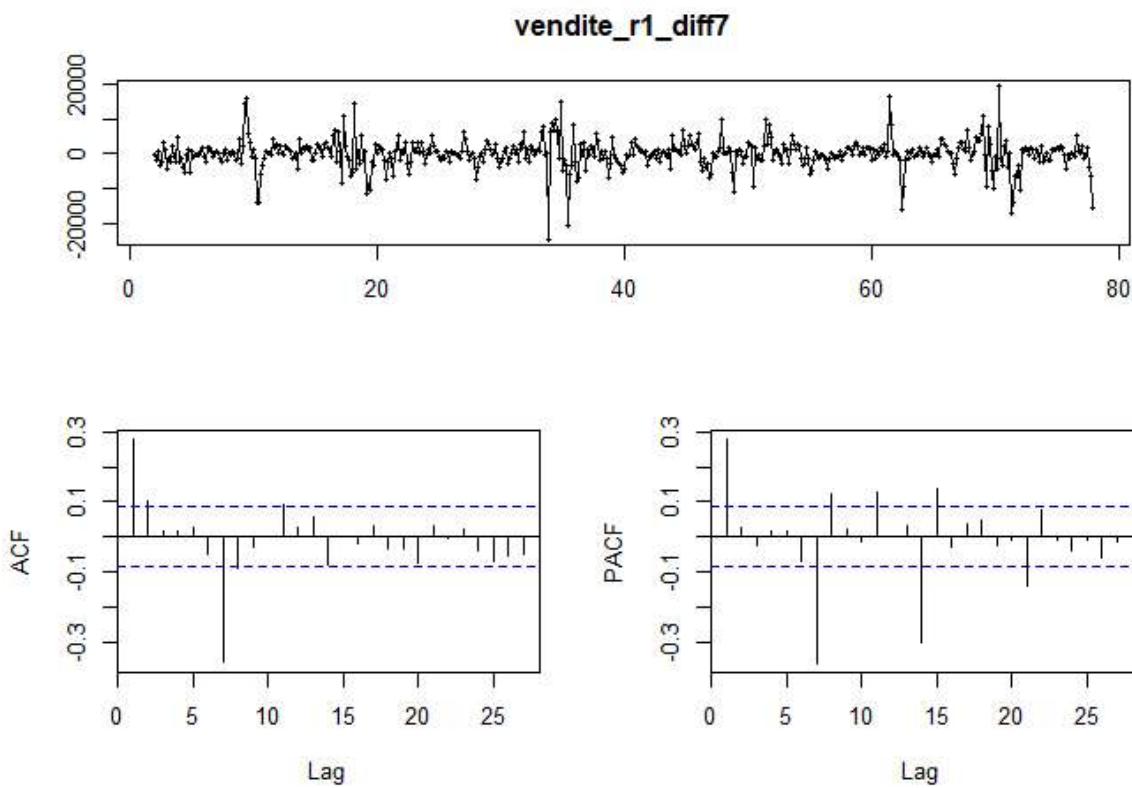
```
## [1] 1
```

```
nsdiffs(vendite_r1_log)
```

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

Il test ci indica la necessità di una differenziazione, sia stagionale che non. Possiamo comunque passare il parametro lambda = "auto" ai successivi modelli qualora volessimo utilizzare i dati trasformati

```
tsdisplay(vendite_r1_diff7)
```



ACF non decade a zero lentamente e non presenta pattern stagionali.

Test

```
# Ljung-Box test
# a non-stationary signal will have a Low p-value
lag.length = 25
Box.test(vendite_r1_diff7, lag=lag.length, type="Ljung-Box") # test stationary signal
```

```
## 
## Box-Ljung test
##
## data: vendite_r1_diff7
## X-squared = 142.99, df = 25, p-value < 2.2e-16
```

```
library(tseries)
# a series with a trend line will have a unit root and result in a Large p-value
adf.test(vendite_r1_diff7, alternative = "stationary")
```

```
## Warning in adf.test(vendite_r1_diff7, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
## 
## Augmented Dickey-Fuller Test
##
## data: vendite_r1_diff7
## Dickey-Fuller = -8.3647, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
# Kwiatkowski-Phillips-Schmidt-Shin test
# a low p-value will indicate a signal that is not trend stationary, has a unit root
kpss.test(vendite_r1_diff7, null="Trend")
```

```
## Warning in kpss.test(vendite_r1_diff7, null = "Trend"): p-value greater than
## printed p-value
```

```
## 
## KPSS Test for Trend Stationarity
##
```

```
## data: vendite_r1_diff7
## KPSS Trend = 0.029478, Truncation lag parameter = 6, p-value = 0.1
```

Il test di Ljung-Box valuta se sono presenti correlazioni significative ai vari ritardi (essendo che una serie storica stazionaria non dovrebbe avere correlazioni significative ai vari lag). Il test, come già notavamo dall'ACF e PACF ci indica che la serie storica non è stazionaria. Questo è dovuto al fatto che non stiamo considerando una stagionalità annuale, la quale viene considerata come una componente che rende la seire storica non stazionaria.

Gli altri test, non ci indicano la presenza di una non stazionarietà della seire storica.

Procediamo a stimare i vari modelli, cercando di catturare i movimenti stagionali annuali attraverso regressori o altri metodi.

Variabili stagionali

```
# Dummy settimanali
require('fastDummies')
dum_day_df <- df[, c("data", "Giorno")]
dum_day_df[, "Giorno"] <- as.factor(dum_day_df$Giorno)
dum_day_df <- dummy_cols(dum_day_df, select_columns = c("Giorno"),
                           # remove_first_dummy = TRUE,
                           remove_selected_columns = TRUE)
dum_day_df <- subset(dum_day_df, select = -c(data,Giorno_Monday))
dum_day_df
```

	Giorno_Friday <int>	Giorno_Saturday <int>	Giorno_Sunday <int>	Giorno_Thursday <int>
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	1
5	1	0	0	0
6	0	1	0	0
7	0	0	1	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0

1-10 of 539 rows | 1-5 of 7 columns

Previous **1** [2](#) [3](#) [4](#) [5](#) [6](#) ... [54](#) [Next](#)

```
# La funzione Arima vuole in input una matrice per i regressori
dum_day <- as.matrix(dum_day_df)
```

```
# trasformo in ts
# train_dumday <- ts(train_dumday, frequency = 7, start = 1, end= 75)
# test_dumday <- ts(test_dumday, frequency = 7, start = c(75,2), end= c(77,7))
```

```
#dummy annuali
dum_ann_df <- read_xlsx("../\\Dati aggiuntivi\\fest_precovid.xlsx", col_types = NULL)
dum_ann_df <- subset(dum_ann_df, select = -date)

dum_ann_df <- as.data.frame(lapply(dum_ann_df, as.integer)) # Trasformo tutte le colonne in
# interi e ri-trasformo in dataframe
head(dum_ann_df)
```

	dec8 <int>	dec24 <int>	dec25 <int>	dec26 <int>	dec31 <int>	jan1 <int>	jan6 <int>	apr25 <int>	mag1 <int>
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

6 rows | 1-10 of 18 columns

```
dum_ann_df <- cbind(dum_ann_df, dum_day_df, df$GASOLIO_RISCALDAMENTO.LITRO, df$Pioggia) #
Aggiungo due regressori considerati utili

colnames(dum_ann_df)[24] = "Riscaldamento" # Rinonimo
colnames(dum_ann_df)[25] = "Pioggia_True" # Rinonimo

dum_ann_df$Pioggia_True[dum_ann_df$Pioggia_True == "True"] <- 1 # Trasformo in dummy
dum_ann_df$Pioggia_True[dum_ann_df$Pioggia_True == ""] <- 0
dum_ann_df$Pioggia_True <- as.integer(dum_ann_df$Pioggia_True) # Trasformo in integer

head(dum_ann_df)
```

	dec8	dec24	dec25	dec26	dec31	jan1	jan6	apr25	mag1
	<int>								
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0

6 rows | 1-10 of 26 columns

```
dum_ann <- as.matrix(dum_ann_df) # Funzione Arima vuole una matrice in input per i regressori
```

```
# trasformo in ts
# train_dumann <- ts(train_dumann, frequency = 7, start = 1, end= 75)
# test_dumann <- ts(test_dumann, frequency = 7, start = c(75,2), end= c(77,7))
```

```
# Fourier terms
# 3 sinusoidi per la stagionalità settimanale
# 15 sinusoidi per la stagionalità annuale
four <- fourier(vendite_r1, K=c(3,15))
four <- as.matrix((four))
```

```
# trasformo in ts
# train_four <- ts(train_four, frequency = 7, start = 1, end= 75)
# test_four <- ts(test_four, frequency = 7, start = c(75,2), end= c(77,7))
```

Stima modelli

Partizionamento dei datasets

```
split_vendite <- ts_split(vendite_r1, sample.out =round(length(vendite_r1)*0.2))
train <- split_vendite$train
test <- split_vendite$test
length(train)
```

```
## [1] 431
```

```
length(test)
```

```
## [1] 108
```

```
# Divisione training-test matrice dummy giornaliere
```

```
train_date <- nrow(dum_day) *0.8  
train_dumday<- dum_day[1:train_date,]  
test_dumday <- dum_day[-c(1:train_date),]
```

```
# Controllo le dimensioni di training e test
```

```
length(train_dumday[,1])
```

```
## [1] 431
```

```
length(test_dumday[,1])
```

```
## [1] 108
```

```
# Divisione training-test dummy annuali
```

```
train_date <- nrow(dum_ann) *0.8  
train_dumann<- dum_ann[1:train_date,]  
test_dumann <- dum_ann[-c(1:train_date),]
```

```
# Controllo le dimensioni di training e test
```

```
length(train_dumday[,1])
```

```
## [1] 431
```

```
length(test_dumday[,1])
```

```
## [1] 108
```

```
# Divisione training-test termini di Fourier
```

```
train_date <- nrow(four) *0.8  
train_four<- four[1:train_date,]  
test_four <- four[-c(1:train_date),]
```

```
# Controllo le dimensioni di training e test  
length(train_four[,1])
```

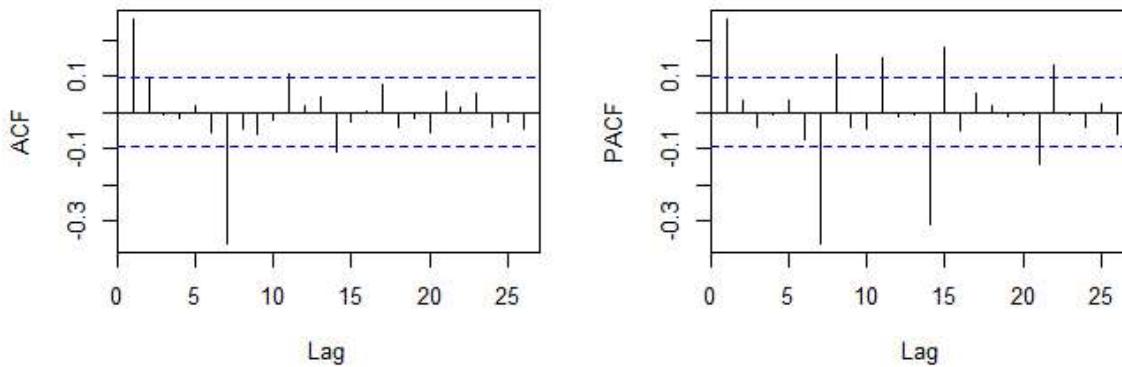
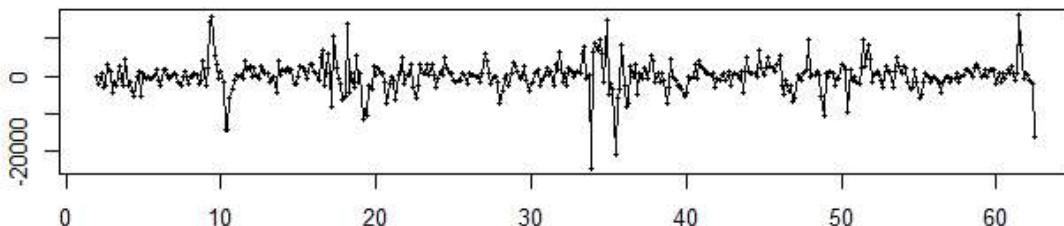
```
## [1] 431
```

```
length(test_four[,1])
```

```
## [1] 108
```

```
tsdisplay(diff(train, 7))
```

diff(train, 7)



Modello 1

- Differenza stagionale di ordine 1
- Componente AR stagionale di ordine 1:
- Componente MA stagionale di ordine 1:
- Componente MA non stagionale di ordine 2
- Componente AR non stagionale di ordine 3
- Costante inclusa

```
mod1 <- Arima(y = train,
                 order = c(3, 0, 2),
                 list(order = c(1, 1, 2)),
                 include.drift = FALSE,
                 lambda = "auto"
               )
```

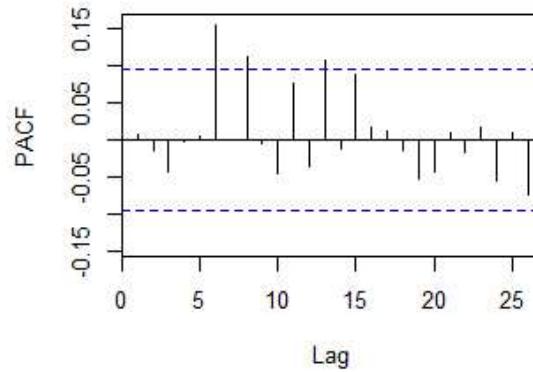
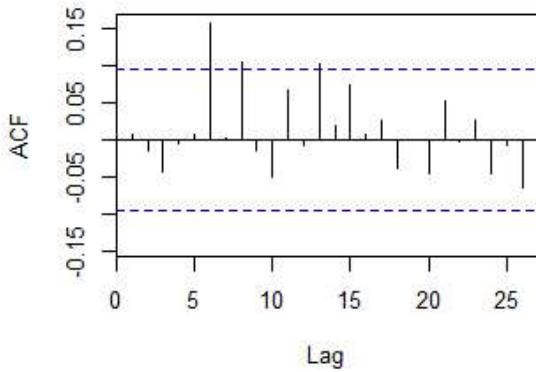
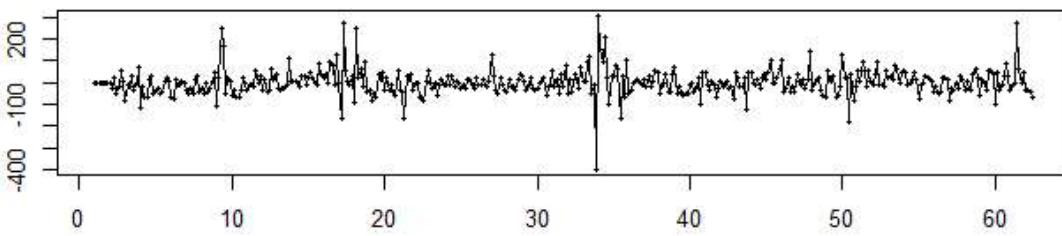
Diagnostica:

```
summary(mod1)
```

```
## Series: train
## ARIMA(3,0,2)(1,1,2)[7]
## Box Cox transformation: lambda= 0.596957
##
## Coefficients:
##             ar1      ar2      ar3      ma1      ma2      sar1      sma1      sma2
##             -0.5857  -0.5677  0.3848  0.9796  0.9986  -0.2848  -0.4611  -0.4218
## s.e.       0.0478   0.0497  0.0466  0.0352  0.0688   0.2255   0.2131   0.1788
##
## sigma^2 = 3647: log likelihood = -2343.93
## AIC=4705.85   AICc=4706.29   BIC=4742.3
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 168.3221 3023.332 1994.99 -2.577323 12.5206 0.7774703 0.02580077
```

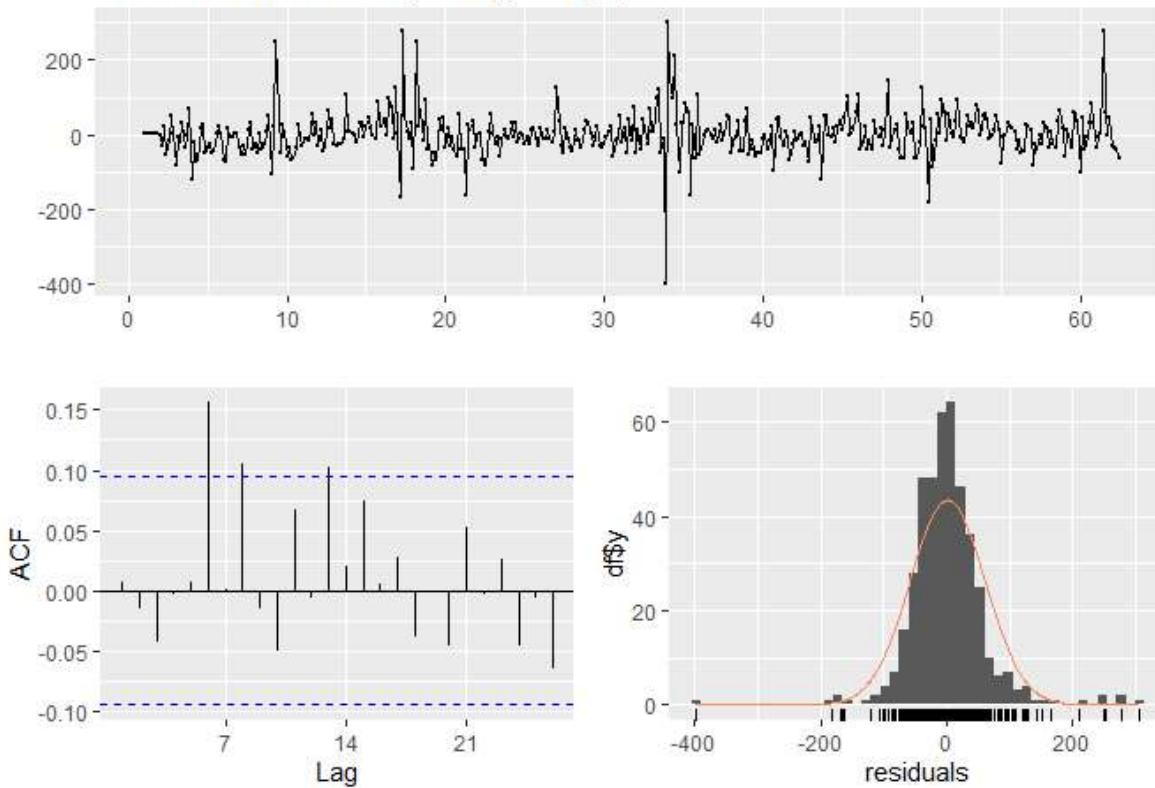
```
tsdisplay(mod1$residuals)
```

mod1\$residuals



```
checkresiduals(mod1, test = FALSE)
```

Residuals from ARIMA(3,0,2)(1,1,2)[7]



```
checkresiduals(mod1, test = "LB", lag = 25, plot = FALSE)
```

```
##  
## Ljung-Box test  
##  
## data: Residuals from ARIMA(3,0,2)(1,1,2)[7]  
## Q* = 31.554, df = 17, p-value = 0.01709  
##  
## Model df: 8. Total lags used: 25
```

```
pars_test(mod1$coef, mod1$var.coef)
```

```
##          ar1        ar2        ar3        ma1        ma2       sar1  
## 0.000000e+00 0.000000e+00 2.220446e-16 0.000000e+00 0.000000e+00 2.065988e-01
```

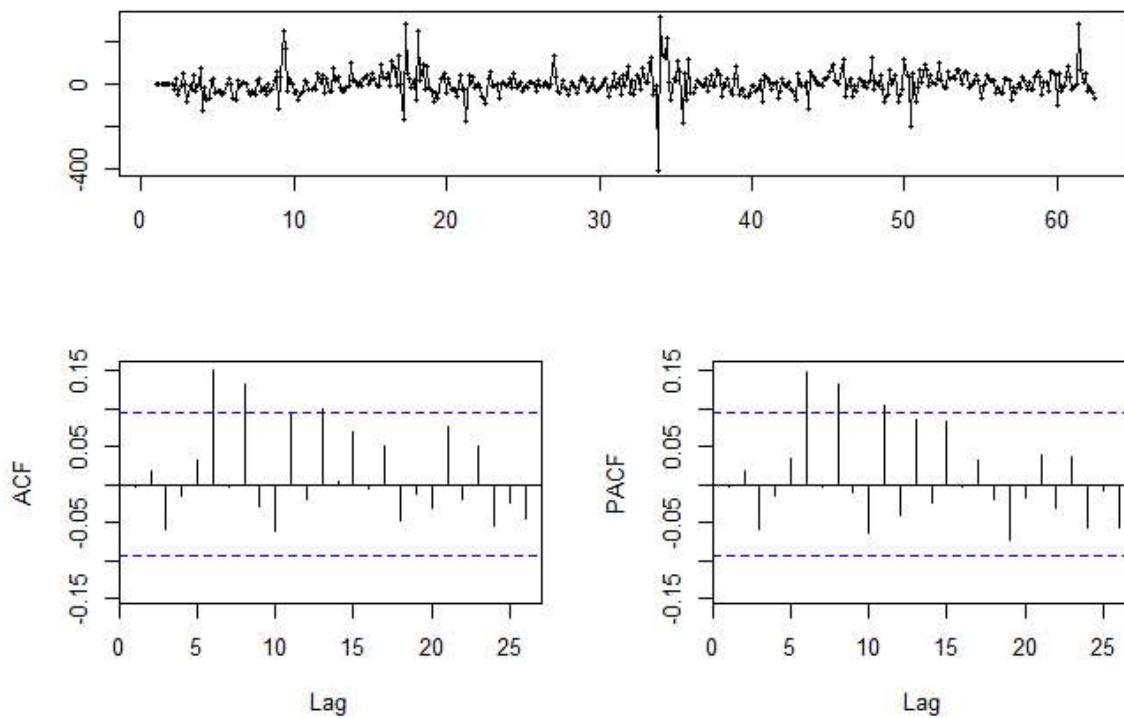
```
##          sma1          sma2
## 3.051162e-02 1.834199e-02
```

```
mod1_auto <- auto.arima(train, lambda = "auto", stepwise = FALSE, approximation = FALSE) #  
Minimizza AIC
```

```
summary(mod1_auto)
```

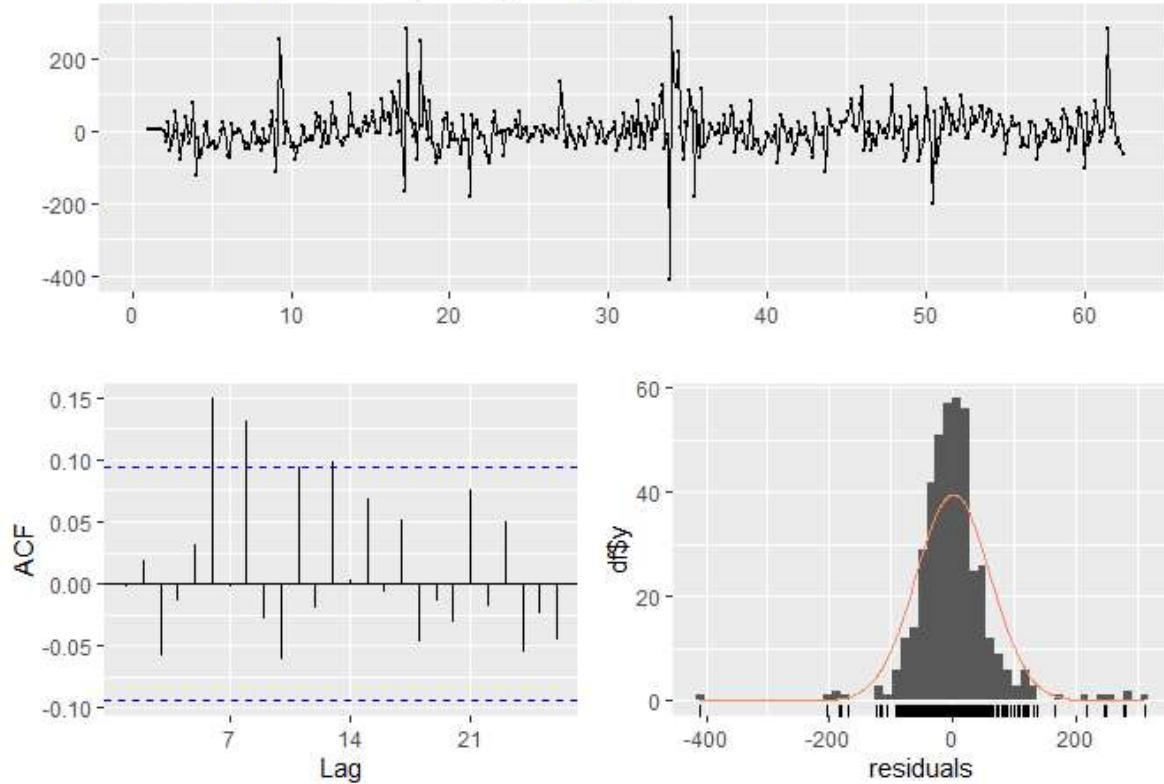
```
## Series: train
## ARIMA(1,0,0)(0,1,2)[7]
## Box Cox transformation: lambda= 0.596957
##
## Coefficients:
##          ar1      sma1      sma2
##        0.3861  -0.7665  -0.1525
## s.e.  0.0464   0.0520   0.0556
##
## sigma^2 = 3710: log likelihood = -2348.71
## AIC=4705.41  AICc=4705.51  BIC=4721.61
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 177.1105 3056.285 2001.647 -2.615921 12.66347 0.7800646 0.02106312
```

```
tsdisplay(mod1_auto$residuals)
```

mod1_auto\$residuals

```
checkresiduals(mod1_auto, test = FALSE)
```

Residuals from ARIMA(1,0,0)(0,1,2)[7]



```
checkresiduals(mod1_auto, test = "LB", lag = 25, plot = FALSE)
```

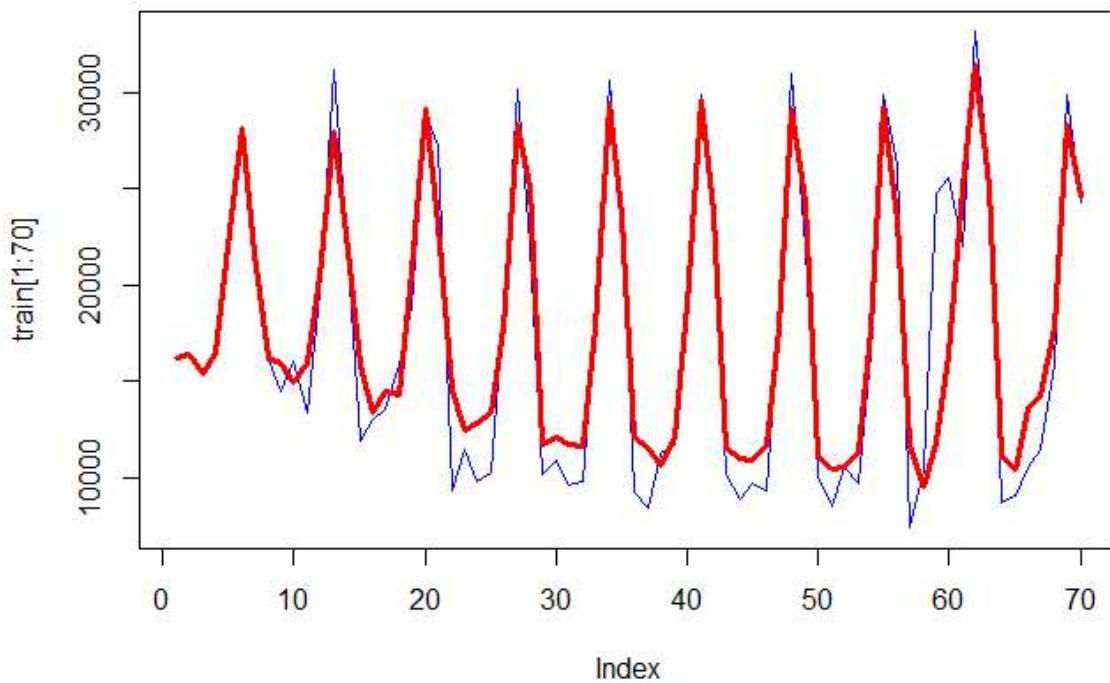
```
##  
## Ljung-Box test  
##  
## data: Residuals from ARIMA(1,0,0)(0,1,2)[7]  
## Q* = 40.419, df = 22, p-value = 0.009654  
##  
## Model df: 3. Total lags used: 25
```

```
parstest(mod1_auto$coef, mod1_auto$var.coef)
```

```
##      ar1      sma1      sma2  
## 0.000000000 0.000000000 0.006062769
```

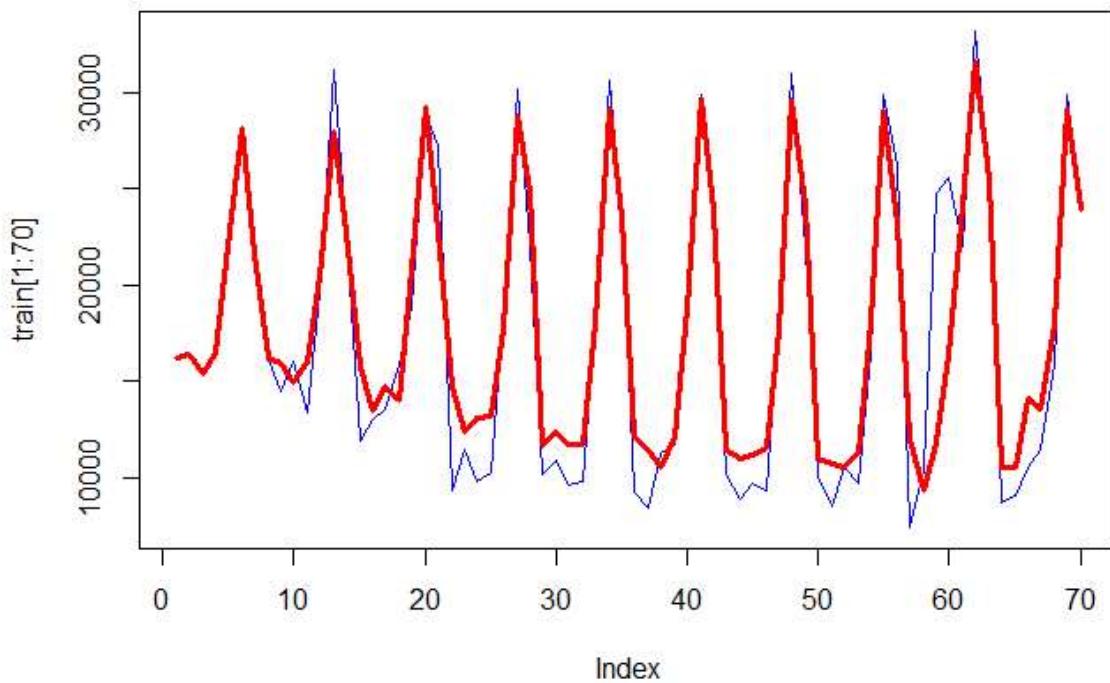
```
# Fit prime 10 settimane  
plot(train[1:70],  
     col = "blue", lwd=0.5,  
     main = "Fitted Values", type = "l")  
lines(mod1$fitted[1:70], col="red", lwd=3)
```

Fitted Values



```
# Fit prime 10 settimane
plot(train[1:70],
      col = "blue", lwd=0.5,
      main = "Fitted Values", type = "l")
lines(mod1_auto$fitted[1:70], col="red", lwd=3)
```

Fitted Values

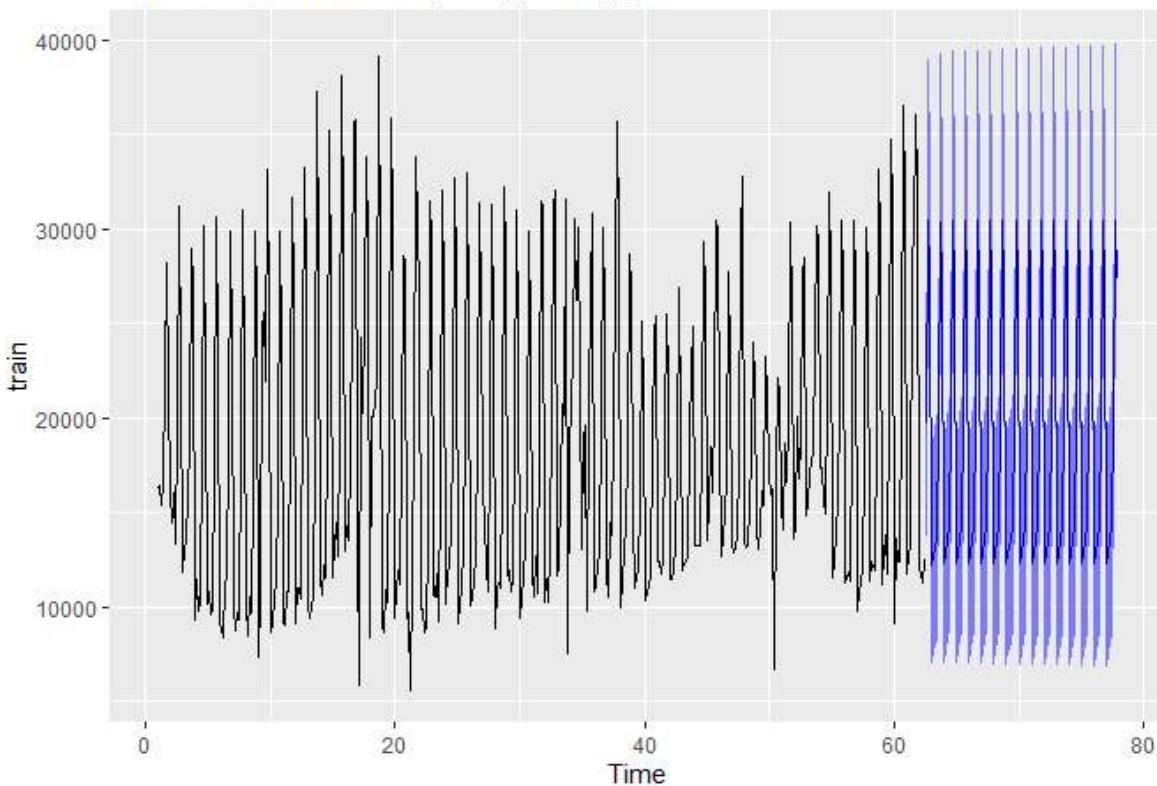


I due modelli presentano caratteristiche molto simili, in termini di erorre di previsione (RMSE, MAE, MAPE) sia in termini di AIC. Il modello *mod1_auto* risulta però avere un numero inferiore di parametri. Si decide quindi di considerare quello.

Previsioni

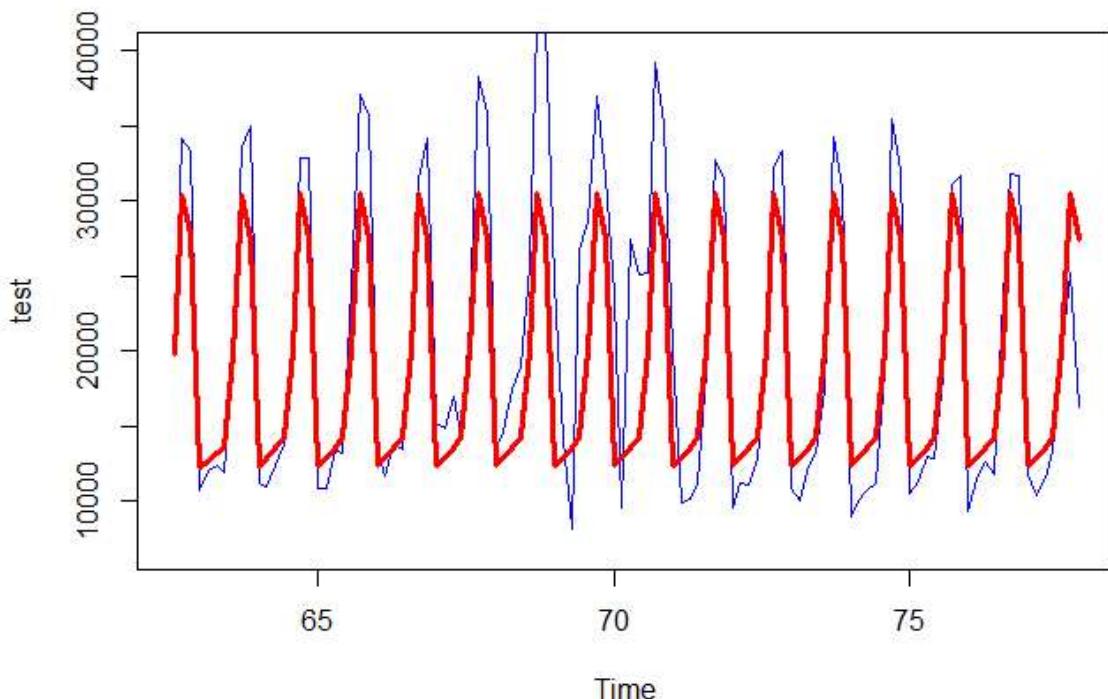
```
# Al momento vengono fatte previsioni 44 passi in avanti (Lunghezza del validation set)
pred_mod1 <- forecast(mod1_auto, h = length(test),
                      level = 95)
```

```
autoplot(pred_mod1)
```

Forecasts from ARIMA(1,0,0)(0,1,2)[7]

```
plot(test,
  col = "blue", lwd=0.5,
  ylim = c(min(pred_mod1$lower), max(pred_mod1$upper)), main = "Predictions")
lines(pred_mod1$mean, col="red", lwd=3)
```

Predictions



```
mape(test, pred_mod1$mean)
```

```
## [1] 17.42545
```

```
mae(test, pred_mod1$mean)
```

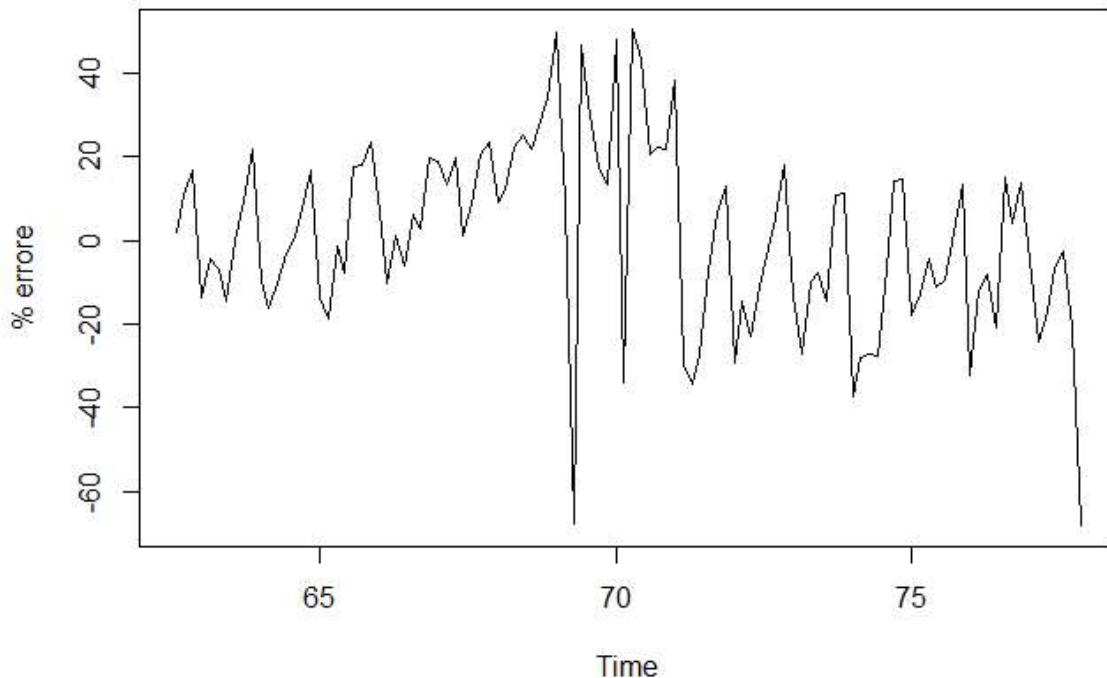
```
## [1] 3584.255
```

```
rmse(test, pred_mod1$mean)
```

```
## [1] 4851.057
```

```
err_plot(test, pred_mod1$mean)
```

Errore percentuale di previsione



Modello 2

Includiamo le dummy giornaliere

```
mod2 <- Arima(y = train,
                 order = c(3, 1, 1),
                 list(order = c(1, 0, 1)),
                 include.constant = TRUE,
                 xreg = train_dumday,
                 lambda = "auto"
               )
```

```
summary(mod2)
```

```
## Series: train
## Regression with ARIMA(3,1,1)(1,0,1)[7] errors
## Box Cox transformation: lambda= 0.596957
##
## Coefficients:
##             ar1      ar2      ar3      ma1      sar1      sma1    drift
##             0.2834 -0.0577 -0.1645 -0.8583  0.8304 -0.7596 -0.0280
```

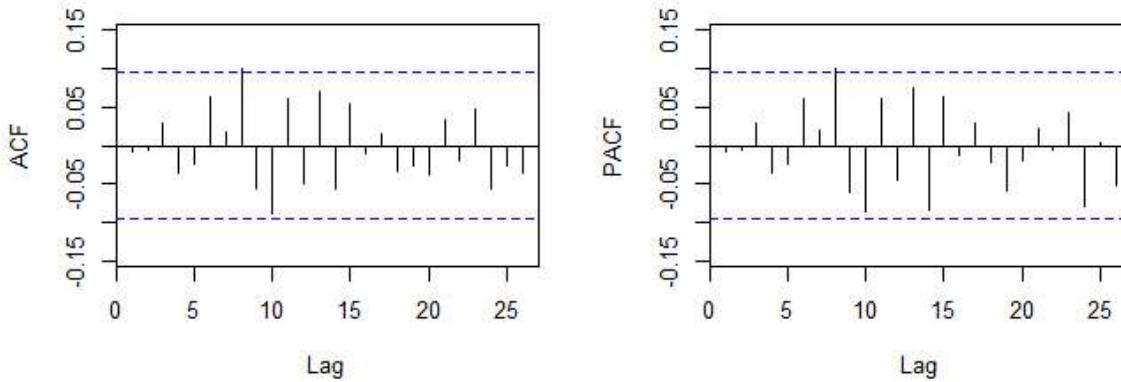
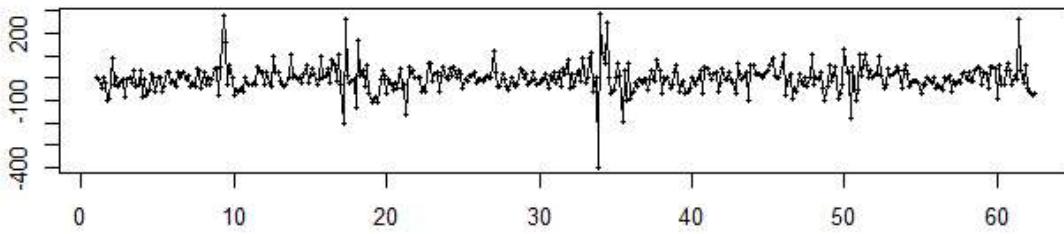
```

## s.e.  0.0591   0.0532   0.0548   0.0404   0.0954   0.1110   0.6055
##      Giorno_Friday  Giorno_Saturday  Giorno_Sunday  Giorno_Thursday
##      160.8718        351.4840       280.7925       41.2722
## s.e.    17.0271       15.3529       12.5805       16.9630
##      Giorno_Tuesday  Giorno_Wednesday
##      8.8474          33.7838
## s.e.    12.5084       15.2843
##
## sigma^2 = 3634: log likelihood = -2366.89
## AIC=4761.78  AICc=4762.79  BIC=4818.67
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 82.58778 3031.756 2030.679 -2.810564 12.76097 0.7913788 0.01340761

```

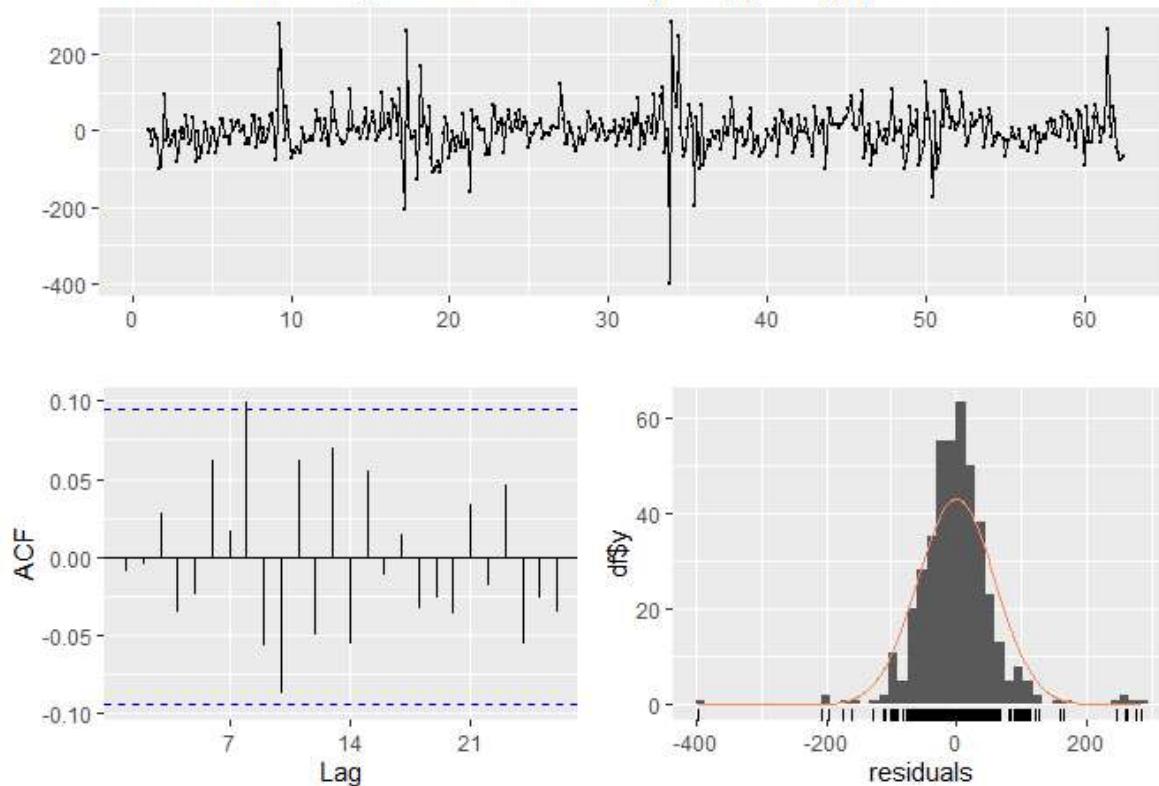
```
tsdisplay(mod2$residuals)
```

mod2\$residuals



```
checkresiduals(mod2, test = FALSE)
```

Residuals from Regression with ARIMA(3,1,1)(1,0,1)[7] errors



```
checkresiduals(mod2, test = "LB", lag = 25, plot = FALSE)
```

```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(3,1,1)(1,0,1)[7] errors  
## Q* = 24.718, df = 19, p-value = 0.17  
##  
## Model df: 6. Total lags used: 25
```

```
pars_test(mod2$coef, mod2$var.coef)
```

```
##          ar1          ar2          ar3          ma1  
## 1.657227e-06 2.778078e-01 2.687810e-03 0.000000e+00  
##          sar1          sma1          drift Giorno_Friday  
## 0.000000e+00 7.690959e-12 9.630957e-01 0.000000e+00  
## Giorno_Saturday Giorno_Sunday Giorno_Thursday Giorno_Tuesday  
## 0.000000e+00 0.000000e+00 1.497151e-02 4.793688e-01
```

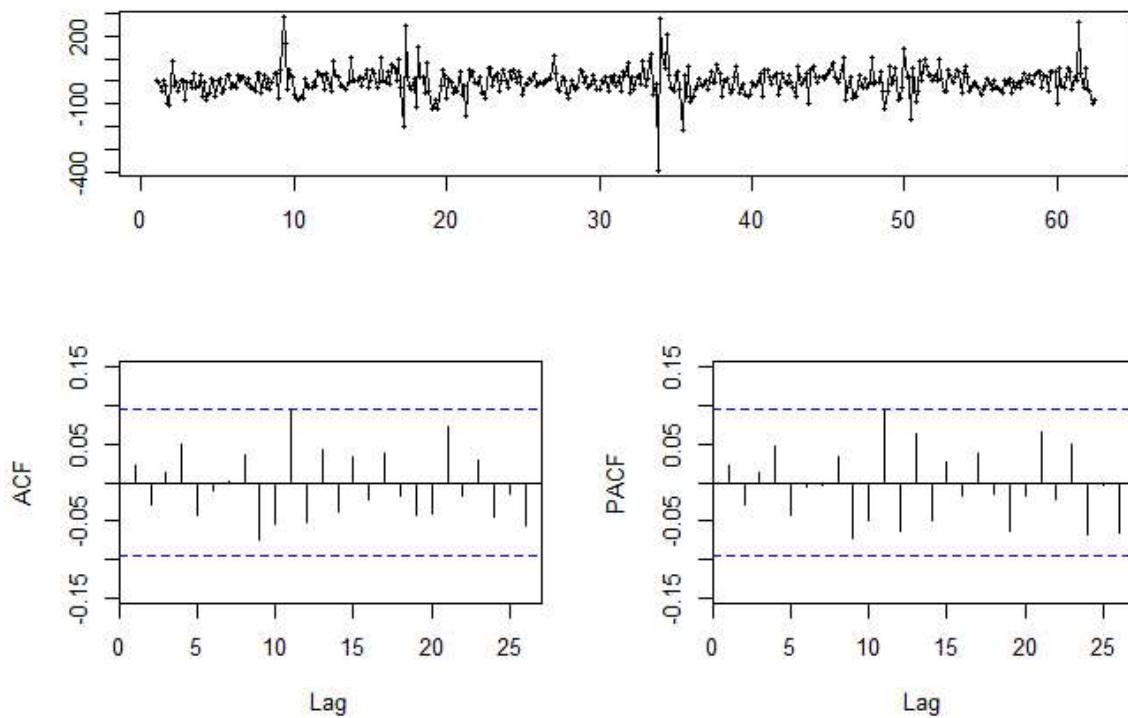
```
## Giorno_Wednesday  
##      2.708083e-02
```

```
mod2_auto <- auto.arima(train, xreg = train_dumday, lambda = "auto", stepwise = FALSE,  
approximation = FALSE)
```

```
summary(mod2_auto)
```

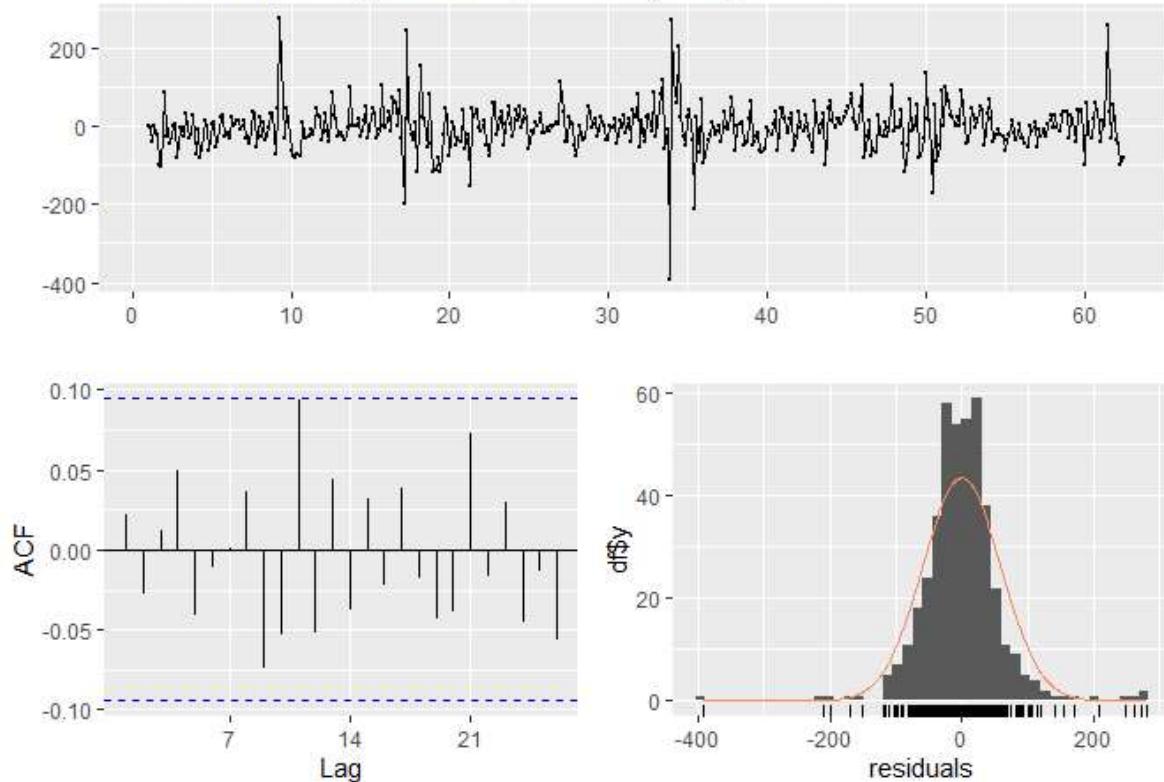
```
## Series: train  
## Regression with ARIMA(2,1,3) errors  
## Box Cox transformation: lambda= 0.596957  
##  
## Coefficients:  
##          ar1      ar2      ma1      ma2      ma3  Giorno_Friday  Giorno_Saturday  
##      1.1036  -0.7931  -1.7146  1.2767  -0.4419       160.1440      351.1014  
## s.e.  0.0766   0.0728   0.1006   0.1653   0.0854       16.9729      14.5358  
##          Giorno_Sunday  Giorno_Thursday  Giorno_Tuesday  Giorno_Wednesday  
##      280.3320        40.5808       8.3452       34.1381  
## s.e.      10.5955        16.9369       10.5506      14.5026  
##  
## sigma^2 = 3576: log likelihood = -2364.46  
## AIC=4752.92  AICc=4753.66  BIC=4801.68  
##  
## Training set error measures:  
##          ME      RMSE      MAE      MPE      MAPE      MASE      ACF1  
## Training set 67.2878 3014.676 2060.132 -2.845048 12.92737 0.8028569 0.04345351
```

```
tsdisplay(mod2_auto$residuals)
```

mod2_auto\$residuals

```
checkresiduals(mod2_auto, test = FALSE)
```

Residuals from Regression with ARIMA(2,1,3) errors



```
checkresiduals(mod2_auto, test = "LB", lag = 25, plot = FALSE)
```

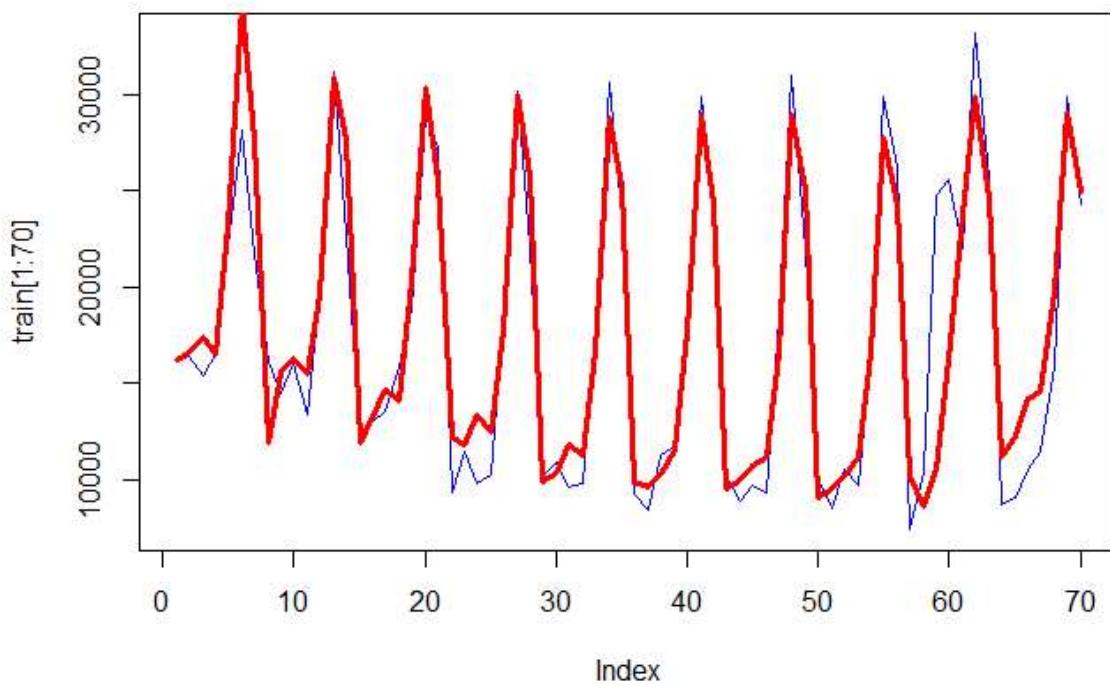
```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(2,1,3) errors  
## Q* = 20.222, df = 20, p-value = 0.4441  
##  
## Model df: 5. Total lags used: 25
```

```
pars_test(mod2_auto$coef, mod2_auto$var.coef)
```

```
##          ar1          ar2          ma1          ma2  
## 0.000000e+00 0.000000e+00 0.000000e+00 1.110223e-14  
##          ma3 Giorno_Friday Giorno_Saturday Giorno_Sunday  
## 2.304602e-07 0.000000e+00 0.000000e+00 0.000000e+00  
## Giorno_Thursday Giorno_Tuesday Giorno_Wednesday  
## 1.657505e-02 4.289645e-01 1.857659e-02
```

```
# Fit prime 10 settimane  
plot(train[1:70],  
     col = "blue", lwd=0.5,  
     main = "Fitted Values", type = "l")  
lines(mod2_auto$fitted[1:70], col="red", lwd=3)
```

Fitted Values

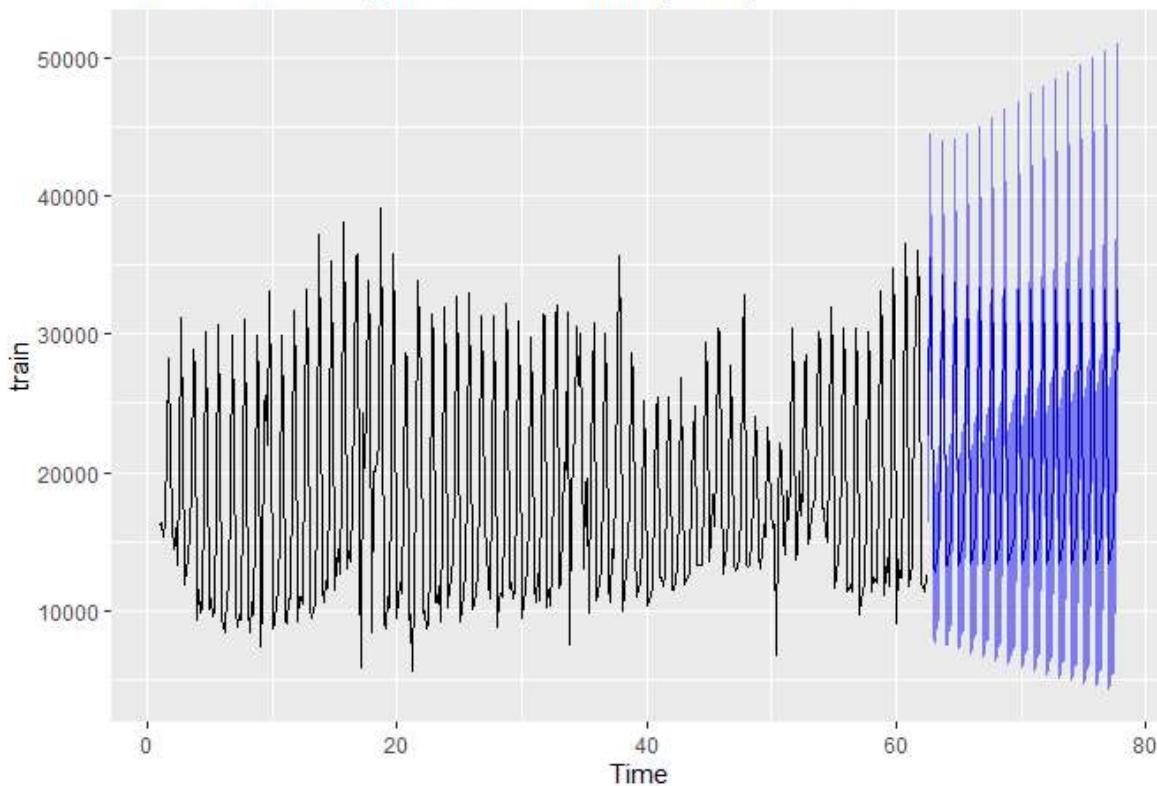


Anche in questo caso i modelli presentano caratteristiche molto simili. Sceglieremo il modello *mod2_auto* in quanto presenta un numero di parametri inferiore

Previsioni

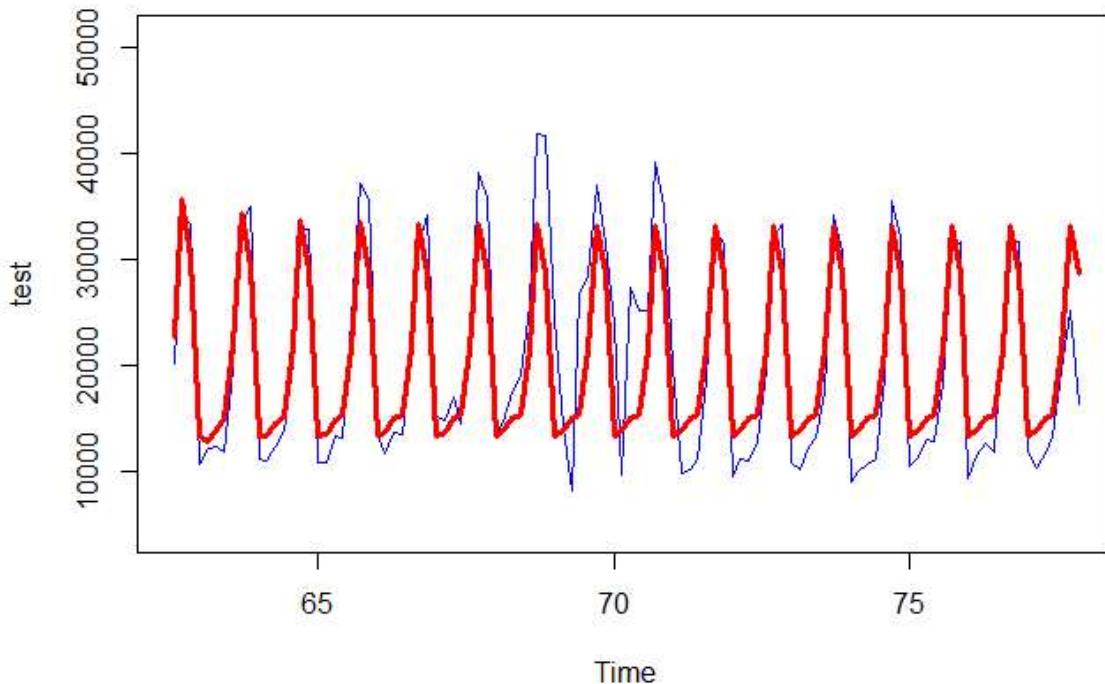
```
pred_mod2 <- forecast(mod2_auto, h = length(test),  
                      level = 95,  
                      xreg = test_dumday)
```

```
autoplot(pred_mod2)
```

Forecasts from Regression with ARIMA(2,1,3) errors

```
plot(test,
  col = "blue", lwd=0.5,
  ylim = c(min(pred_mod2$lower), max(pred_mod2$upper)), main = "Predictions")
lines(pred_mod2$mean, col="red", lwd=3)
```

Predictions



```
mape(test, pred_mod2$mean)
```

```
## [1] 19.71614
```

```
mae(test, pred_mod2$mean)
```

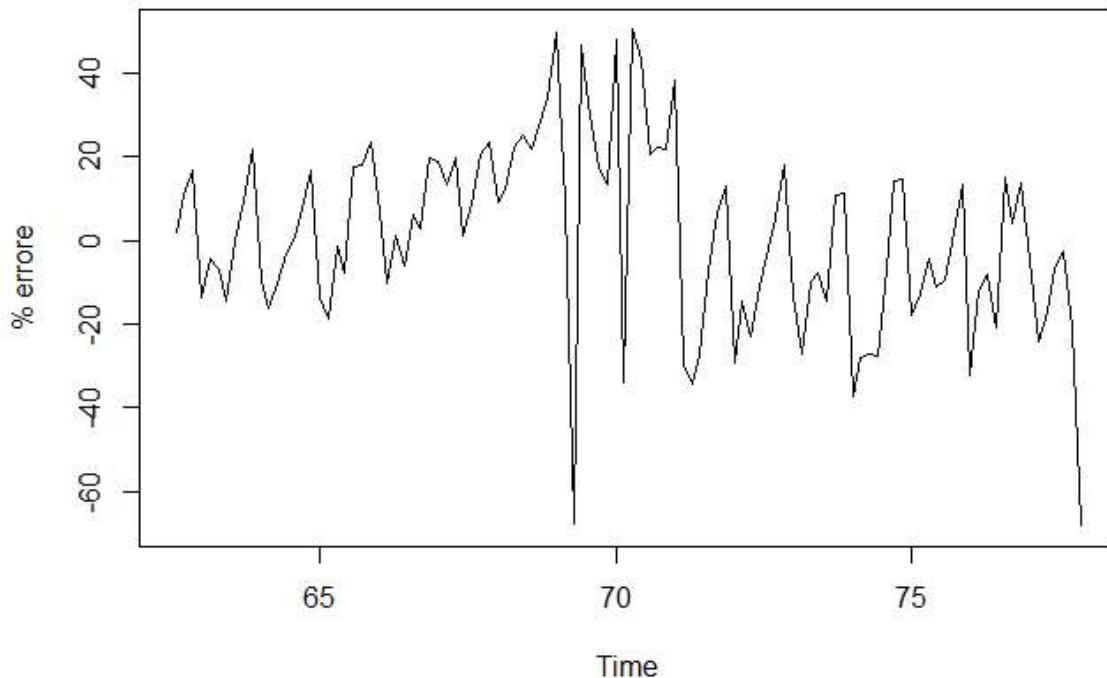
```
## [1] 3513.682
```

```
rmse(test, pred_mod2$mean)
```

```
## [1] 4444.286
```

```
err_plot(test, pred_mod1$mean)
```

Errore percentuale di previsione



Modello 3

Come regressori vengono usati i termini di fourier: 3 armoniche per la stagionalità settimanale e 15 per la stagionalità annuale

```
mod3 <- Arima(y = train,
                 order = c(3, 1, 2),
                 list(order = c(1, 0, 1)),
                 include.constant = TRUE,
                 xreg = train_four,
                 lambda = "auto"
               )
```

```
summary(mod3)
```

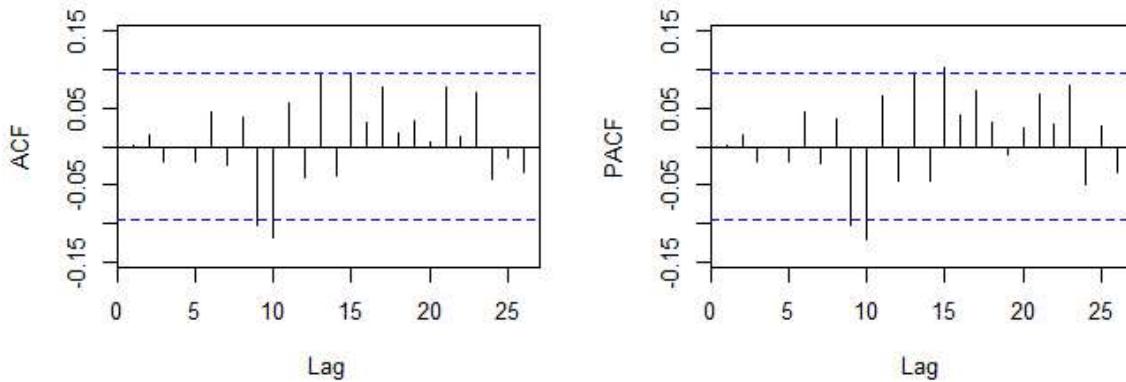
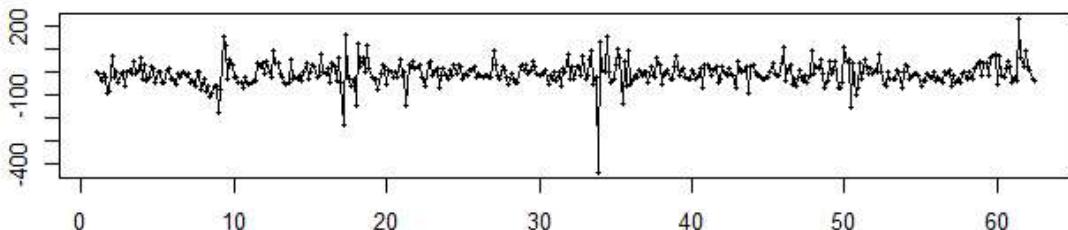
```
## Series: train
## Regression with ARIMA(3,1,2)(1,0,1)[7] errors
## Box Cox transformation: lambda= 0.596957
##
## Coefficients:
##             ar1      ar2      ar3      ma1      ma2      sar1      sma1     drift
##
```

```

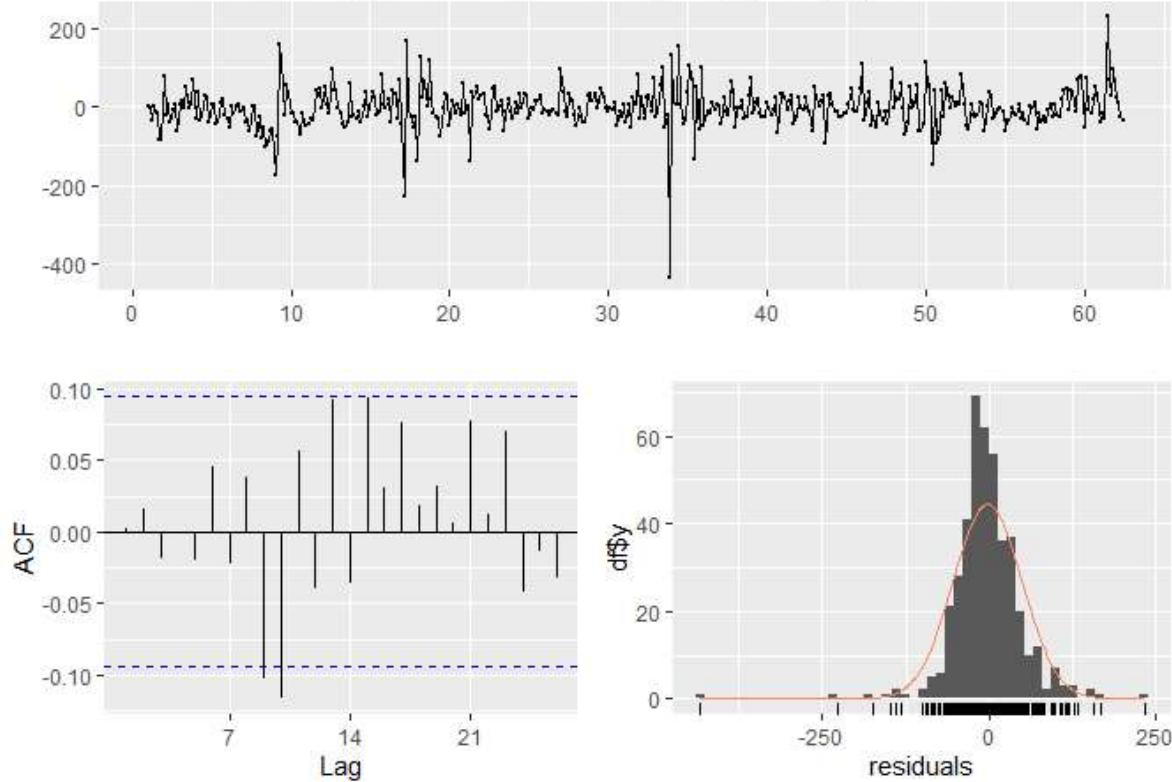
##      0.9465 -0.2950 -0.0948 -1.8116  0.8116  0.8375 -0.7686  0.1104
## s.e.  0.0737  0.0676  0.0561  0.0622  0.0620  0.0872  0.1016  0.0135
##      S1-7    C1-7    S2-7    C2-7    S3-7    C3-7    S1-365
##     -121.7212 112.7873 -76.9965 27.6287 -11.5960 15.0541 11.1800
## s.e.   8.0294   8.0186   4.5804   4.5866   4.1034   4.1165   2.2579
##      C1-365  S2-365  C2-365  S3-365  C3-365  S4-365  C4-365  S5-365
##     -1.2263  0.6472 -4.8556 -14.7253 20.2975  6.4026 -2.4100  9.4336
## s.e.  2.0721  1.7979  1.8398  1.7240  1.7097  1.6987  1.6949  1.7089
##      C5-365  S6-365  C6-365  S7-365  C7-365  S8-365  C8-365  S9-365
##     -6.6087 -5.6698 24.2727 13.4757  9.0475  2.9281  4.5260 -4.4077
## s.e.  1.7194  1.7538  1.7441  1.7951  1.8003  1.8503  1.8677  1.9359
##      C9-365  S10-365 C10-365 S11-365 C11-365 S12-365 C12-365 S13-365
##     2.2059  4.7801 -2.8917 -2.3401  1.1393 -2.9514  2.3021  8.6969
## s.e.  1.9327  2.0176  2.0243  2.1088  2.1216  2.2148  2.2132  2.3099
##      C13-365 S14-365 C14-365 S15-365 C15-365
##     4.5468  2.5462  5.1502 -7.8058 -7.6728
## s.e.  2.3221  2.4180  2.4229  2.5276  2.5233
##
## sigma^2 = 2959: log likelihood = -2309.34
## AIC=4708.68  AICc=4719.46  BIC=4891.55
##
## Training set error measures:
##          ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set -43.34081 2675.203 1808.782 -3.13757 11.37821 0.7049029 0.03344331

```

```
tsdisplay(mod3$residuals)
```

mod3\$residuals

```
checkresiduals(mod3, test = FALSE)
```

Residuals from Regression with ARIMA(3,1,2)(1,0,1)[7] errors

```
checkresiduals(mod3, test = "LB", lag = 25, plot = FALSE)
```

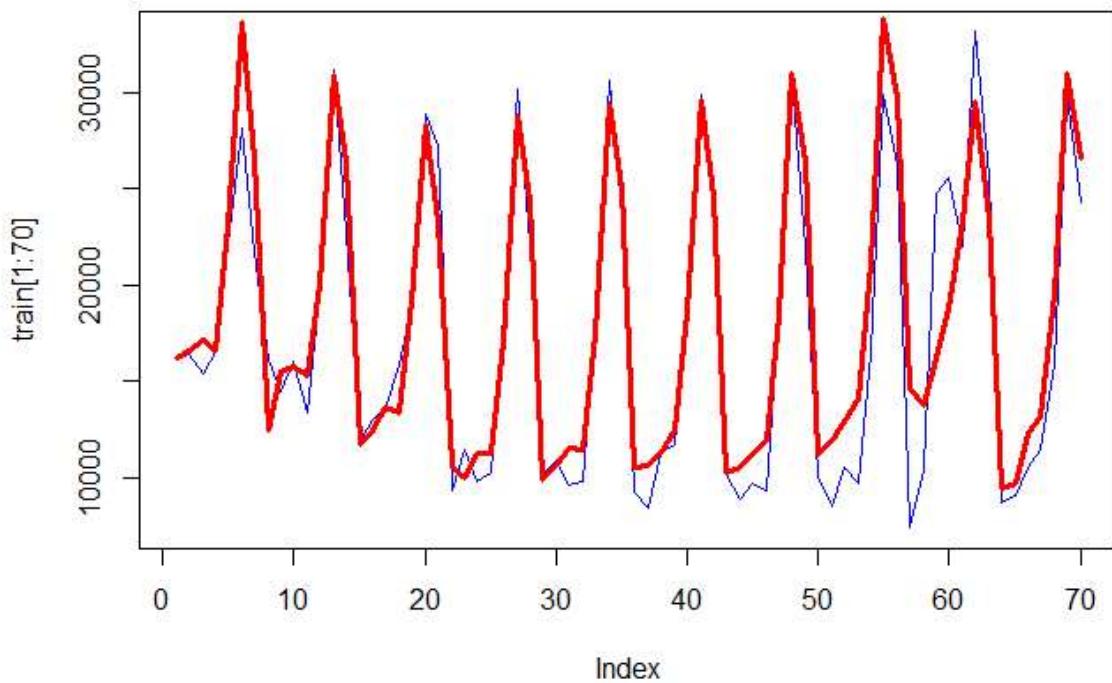
```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(3,1,2)(1,0,1)[7] errors  
## Q* = 33.105, df = 18, p-value = 0.01621  
##  
## Model df: 7. Total lags used: 25
```

```
pars_test(mod3$coef, mod3$var.coef)
```

```
##      ar1        ar2        ar3        ma1        ma2       sar1  
## 0.000000e+00 1.282314e-05 9.137857e-02 0.000000e+00 0.000000e+00 0.000000e+00  
##      sma1       drift      S1-7      C1-7      S2-7      C2-7  
## 3.930190e-14 2.220446e-16 0.000000e+00 0.000000e+00 0.000000e+00 1.704004e-09  
##      S3-7      C3-7      S1-365     C1-365     S2-365     C2-365  
## 4.713484e-03 2.551541e-04 7.361066e-07 5.539675e-01 7.188396e-01 8.308080e-03  
##      S3-365     C3-365     S4-365     C4-365     S5-365     C5-365  
## 0.000000e+00 0.000000e+00 1.638602e-04 1.550586e-01 3.383369e-08 1.212348e-04  
##      S6-365     C6-365     S7-365     C7-365     S8-365     C8-365  
## 1.225301e-03 0.000000e+00 6.061818e-14 5.020989e-07 1.135402e-01 1.538173e-02  
##      S9-365     C9-365     S10-365    C10-365    S11-365    C11-365  
## 2.279717e-02 2.537268e-01 1.782478e-02 1.531465e-01 2.671331e-01 5.912683e-01  
##      S12-365    C12-365    S13-365    C13-365    S14-365    C14-365  
## 1.826828e-01 2.982613e-01 1.665524e-04 5.021979e-02 2.923215e-01 3.353128e-02  
##      S15-365    C15-365  
## 2.013873e-03 2.360039e-03
```

```
# Fit prime 10 settimane  
plot(train[1:70],  
      col = "blue", lwd=0.5,  
      main = "Fitted Values", type = "l")  
lines(mod3$fitted[1:70], col="red", lwd=3)
```

Fitted Values



```
# Attenzione a runnare, ci mette un po' !
# mod3_auto <- auto.arima(train, xreg = train_four, Lambda = "auto", stepwise = FALSE,
#                           approximation = FALSE)
```

```
# summary(mod3_auto)
```

```
# tsdisplay(mod3_auto$residuals)
```

```
# checkresiduals(mod3_auto, test = FALSE)
```

```
# checkresiduals(mod3_auto, test = "LB", lag = 25, plot = FALSE)
```

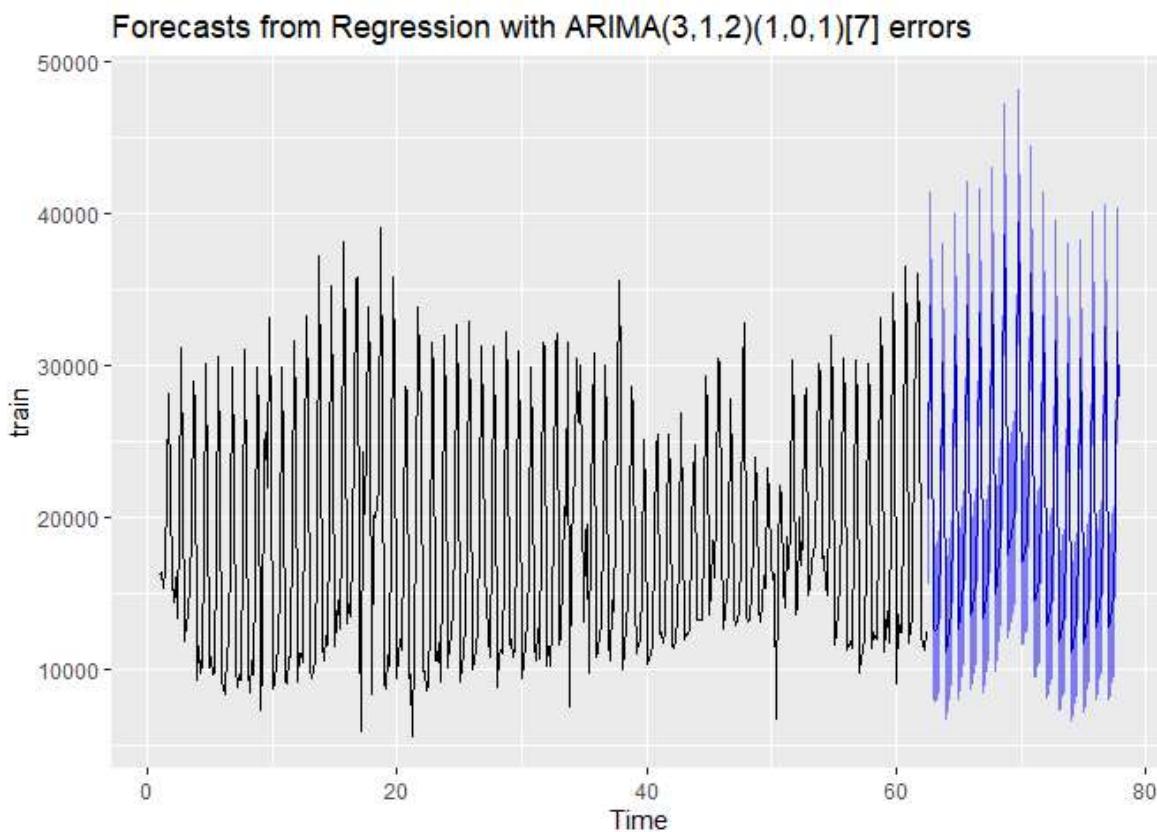
```
# pars_test(mod3_auto$coef, mod3_auto$var.coef)
```

```
# Fit prime 10 settimane
# plot(train[1:70],
#       col = "blue", lwd=0.5,
#       main = "Fitted Values", type = "l")
#lines(mod3_auto$fitted[1:70], col="red", lwd=3)
```

Scelgo il modello *mod3* in quanto presenta caratteristiche migliori per quanto riguarda errori di previsioni e residui, anche se risulta essere meno parsimonioso

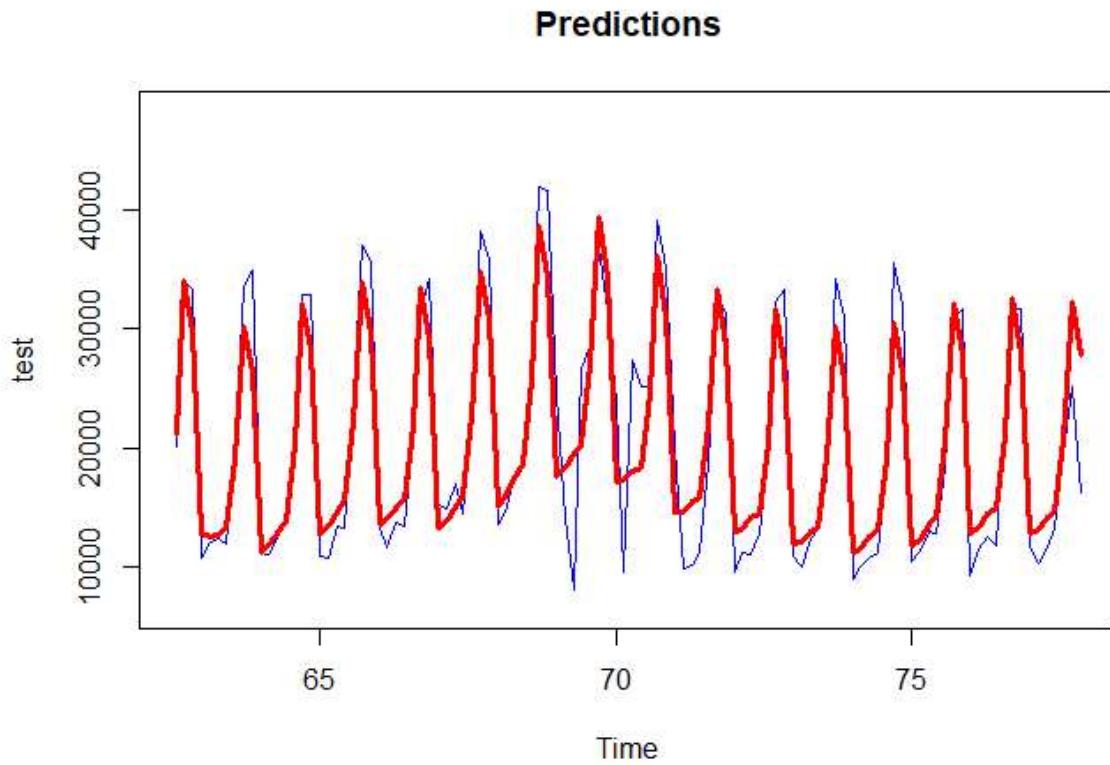
```
pred_mod3 <- forecast(mod3, h = length(test),
                       level = 95,
                       xreg = test_four
                     )
```

```
autoplot(pred_mod3)
```



```
plot(test,
      col = "blue", lwd=0.5,
```

```
ylim = c(min(pred_mod3$lower), max(pred_mod3$upper)), main = "Predictions")
lines(pred_mod3$mean, col="red", lwd=3)
```



```
mape(test, pred_mod3$mean)
```

```
## [1] 15.72596
```

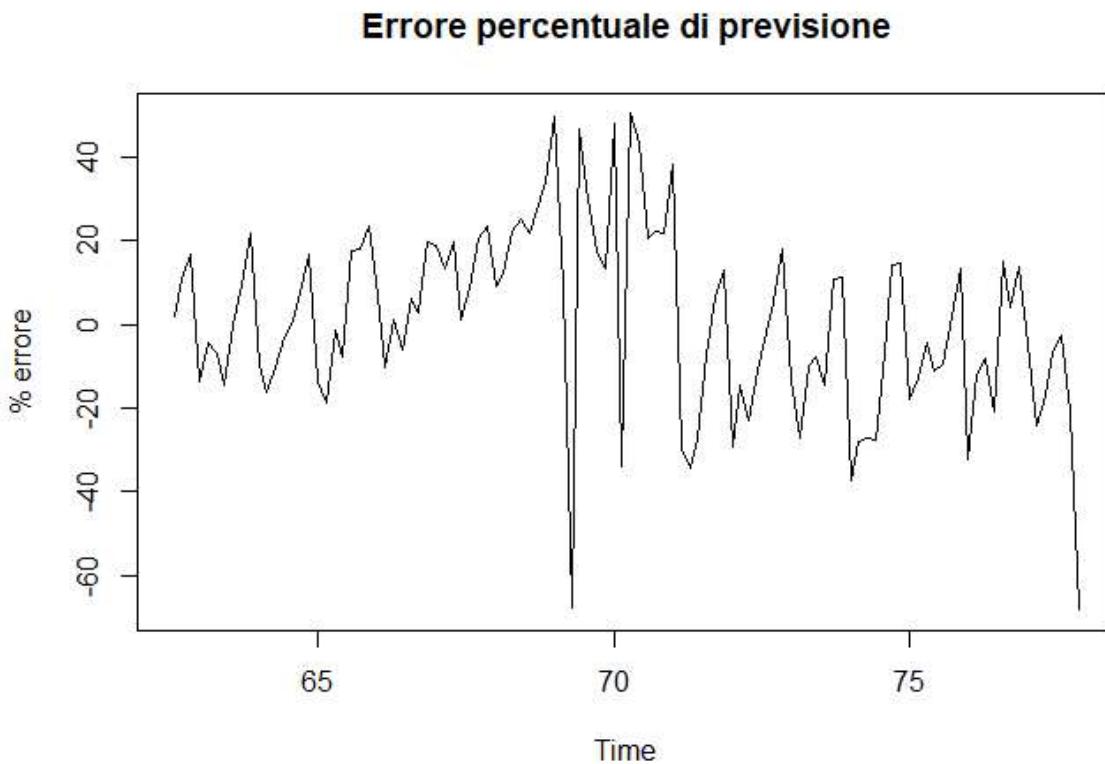
```
mae(test, pred_mod3$mean)
```

```
## [1] 2767.864
```

```
rmse(test, pred_mod3$mean)
```

```
## [1] 3680.758
```

```
err_plot(test, pred_mod1$mean)
```



Modello 4

Includiamo le dummy giornaliere e quelle annuali

```
mod4 <- Arima(y = train,
                 order = c(3, 1, 3),
                 list(order = c(1, 0, 0)),
                 include.constant = TRUE,
                 xreg = train_dumann,
                 lambda = "auto"
               )
```

```
summary(mod4)
```

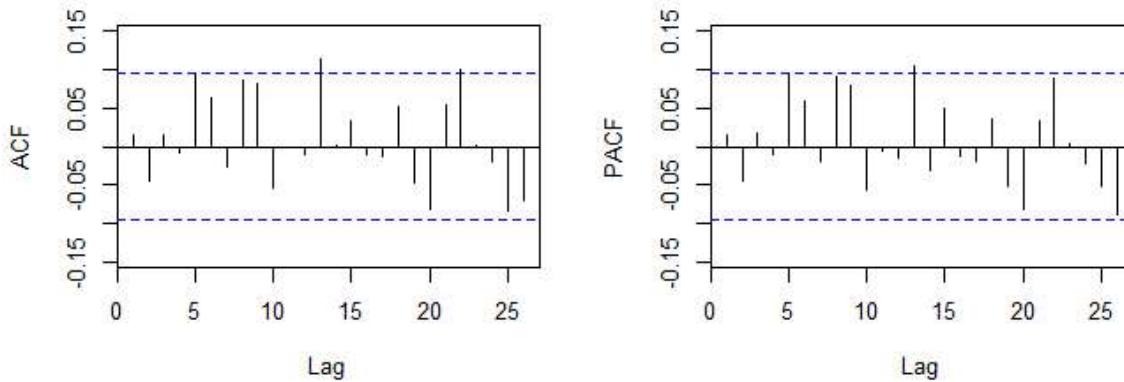
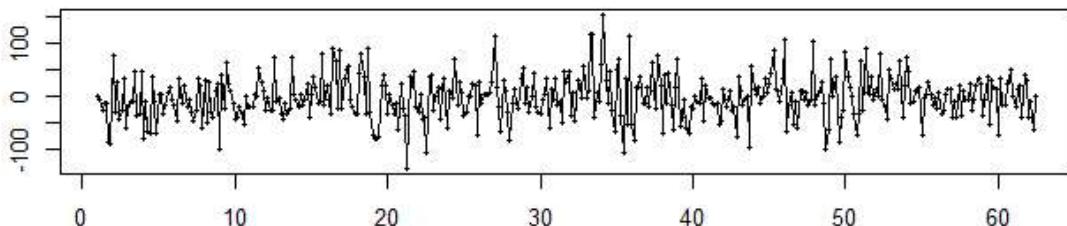
```
## Series: train
## Regression with ARIMA(3,1,3)(1,0,0)[7] errors
## Box Cox transformation: lambda= 0.596957
```

```

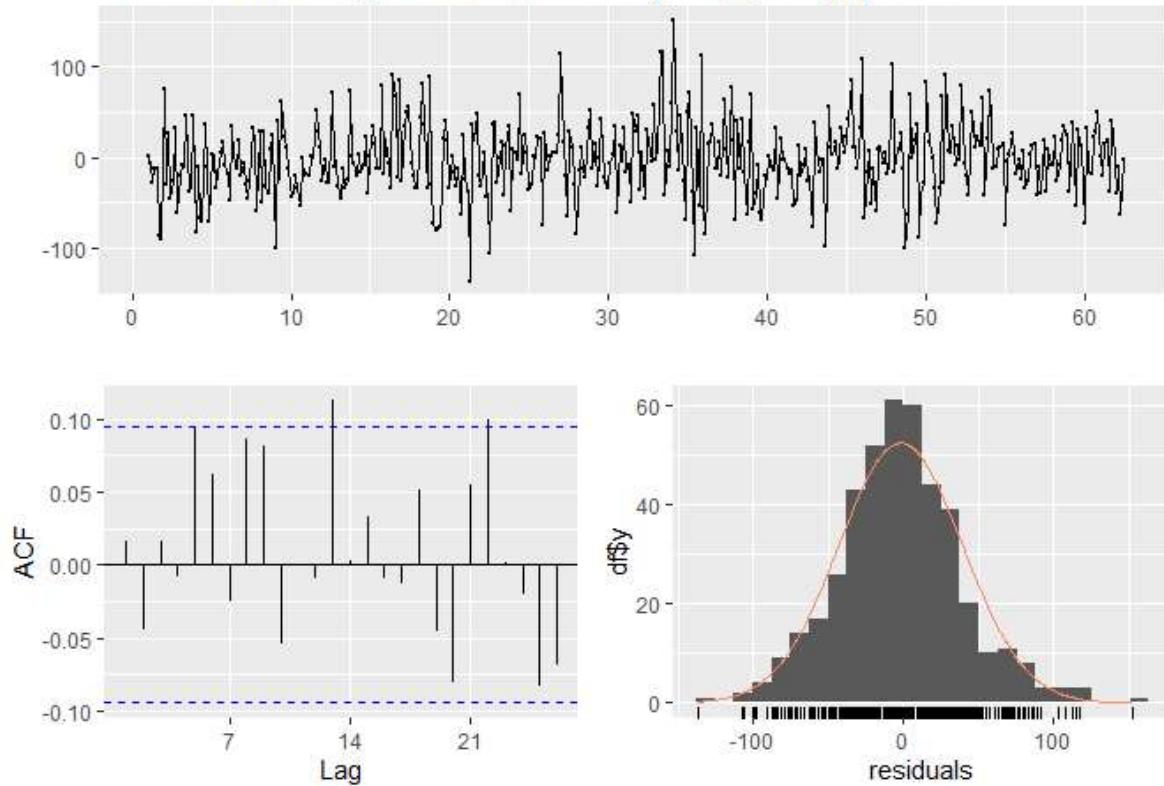
## 
## Coefficients:
##             ar1      ar2      ar3      ma1      ma2      ma3     sar1    drift
##         -1.4223  -0.1750  0.3489  0.8469  -0.8983  -0.9486  0.4579  0.0684
##  s.e.     0.0617  0.0933  0.0610  0.0343   0.0185   0.0278  0.0486  0.0479
##             dec8     dec24     dec25     dec26     dec31     jan1     jan6
##        -16.2391 -12.0893 -251.3793 129.9935 -69.4739  94.5891 -15.6015
##  s.e.    37.2227  41.3777  43.9516  38.9521  41.0362  41.2503  36.0212
##             apr25     mag1     jun2     aug15     oct31     nov1   eastsun
##        239.5069  88.1197 -30.0892 -203.4270  277.4561 199.1710 -399.1737
##  s.e.    37.7997  37.1441  35.9462  35.2313  26.9413  27.8332  38.4545
##             eastermon   martgrasso   bridge Giorno_Friday Giorno_Saturday
##          59.0976     2.6155  43.3480      155.4125      349.4683
##  s.e.    38.5960    36.9686  17.2843      13.9317      13.6290
##             Giorno_Sunday Giorno_Thursday Giorno_Tuesday Giorno_Wednesday
##         285.2473      31.9211      13.3480      25.5025
##  s.e.    10.6806     13.8891     10.6151     13.5876
##             Riscaldamento Pioggia_True
##          -580.0398     18.5526
##  s.e.    217.7146     4.6595
## 
## sigma^2 = 1842: log likelihood = -2212.46
## AIC=4492.91  AICc=4498.94  BIC=4631.08
## 
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -47.82024 2155.133 1630.344 -2.050682 9.825467 0.6353636
##               ACF1
## Training set 0.001611994

```

```
tsdisplay(mod4$residuals)
```

mod4\$residuals

```
checkresiduals(mod4, test = FALSE)
```

Residuals from Regression with ARIMA(3,1,3)(1,0,0)[7] errors

```
checkresiduals(mod4, test = "LB", lag = 25, plot = FALSE)
```

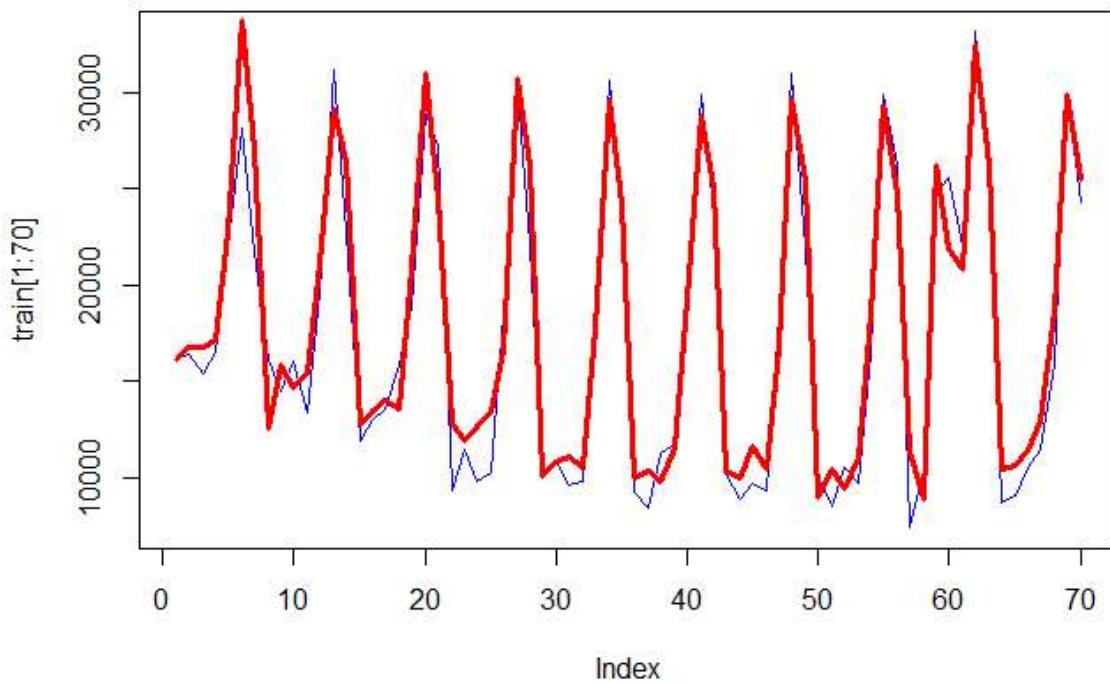
```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(3,1,3)(1,0,0)[7] errors  
## Q* = 35.068, df = 18, p-value = 0.009267  
##  
## Model df: 7. Total lags used: 25
```

```
pars_test(mod4$coef, mod4$var.coef)
```

	ar1	ar2	ar3	ma1
##	0.000000e+00	6.078466e-02	1.051515e-08	0.000000e+00
##	ma2	ma3	sar1	drift
##	0.000000e+00	0.000000e+00	0.000000e+00	1.528943e-01
##	dec8	dec24	dec25	dec26
##	6.626416e-01	7.701564e-01	1.068671e-08	8.460749e-04
##	dec31	jan1	jan6	apr25
##	9.045741e-02	2.184497e-02	6.649275e-01	2.354781e-10
##	mag1	jun2	aug15	oct31
##	1.767430e-02	4.025569e-01	7.739059e-09	0.000000e+00
##	nov1	eastsun	eastermon	martgrasso
##	8.313350e-13	0.000000e+00	1.257234e-01	9.435977e-01
##	bridge	Giorno_Friday	Giorno_Saturday	Giorno_Sunday
##	1.214387e-02	0.000000e+00	0.000000e+00	0.000000e+00
##	Giorno_Thursday	Giorno_Tuesday	Giorno_Wednesday	Riscaldamento
##	2.154588e-02	2.085882e-01	6.053281e-02	7.716680e-03
##	Pioggia_True			
##	6.843326e-05			

```
# Fit prime 10 settimane
plot(train[1:70],
      col = "blue", lwd=0.5,
      main = "Fitted Values", type = "l")
lines(mod4$fitted[1:70], col="red", lwd=3)
```

Fitted Values



```
# mod4_auto <- auto.arima(train, xreg = train_dumann, Lambda = "auto", stepwise = FALSE,
approximation = FALSE)
```

```
# summary(mod4_auto)
```

```
# tsdisplay(mod4_auto$residuals)
```

```
# checkresiduals(mod4_auto, test = FALSE)
```

```
# checkresiduals(mod4_auto, test = "LB", lag = 25, plot = FALSE)
```

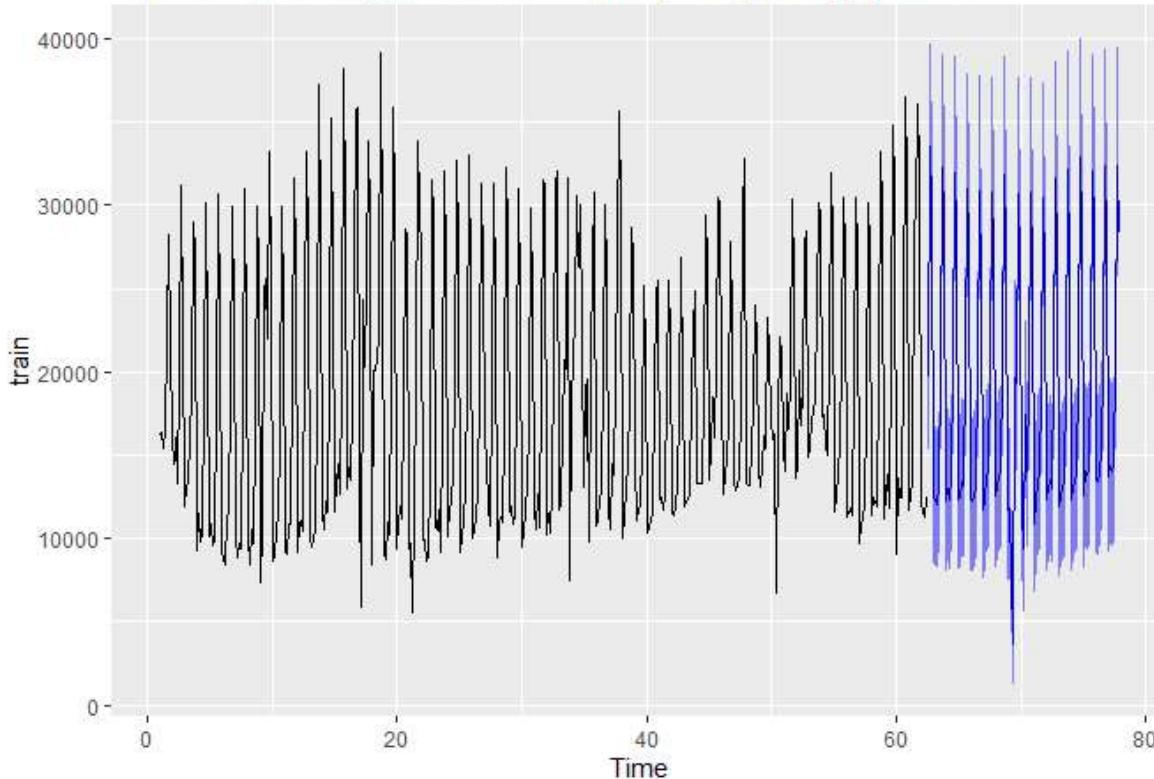
```
# pars_test(mod4_auto$coef, mod4_auto$var.coef)
```

Scegiamo il modello *mod4* in quanto presenta caratteristiche migliori in termini di previsioni e residui.

```
pred_mod4 <- forecast(mod4, h = length(test),
                      level = 95,
                      xreg = test_dumann
)
```

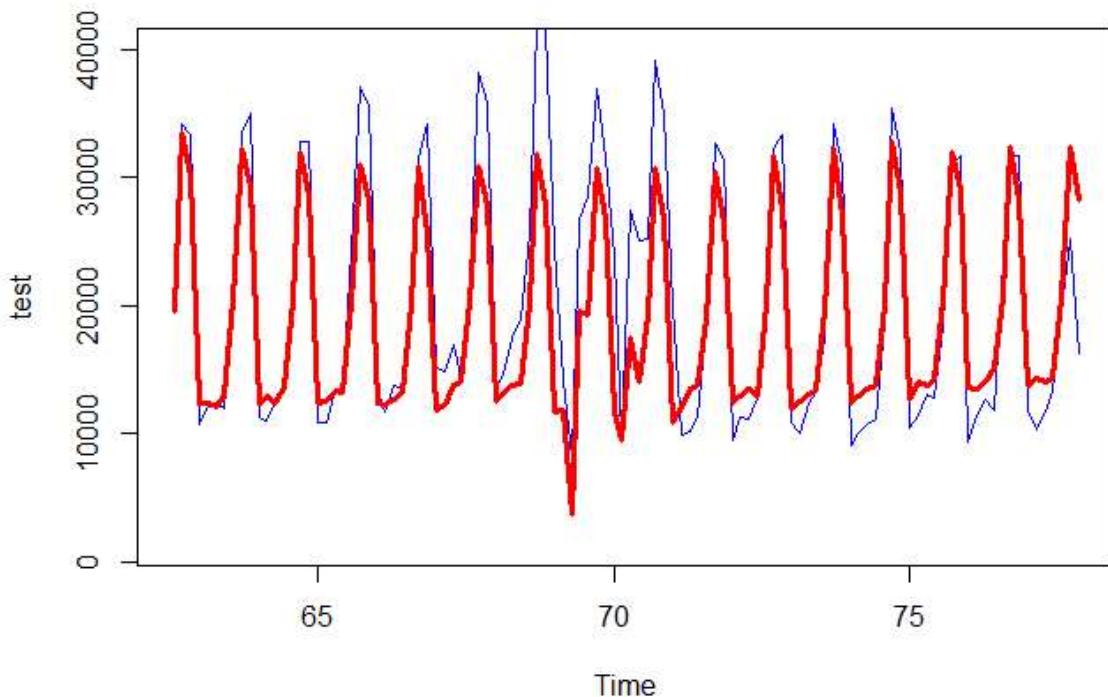
```
autoplot(pred_mod4)
```

Forecasts from Regression with ARIMA(3,1,3)(1,0,0)[7] errors



```
plot(test,
      col = "blue", lwd=0.5,
      ylim = c(min(pred_mod4$lower), max(pred_mod4$upper)), main = "Predictions")
lines(pred_mod4$mean, col="red", lwd=3)
```

Predictions



```
mape(test, pred_mod4$mean)
```

```
## [1] 16.46479
```

```
mae(test, pred_mod4$mean)
```

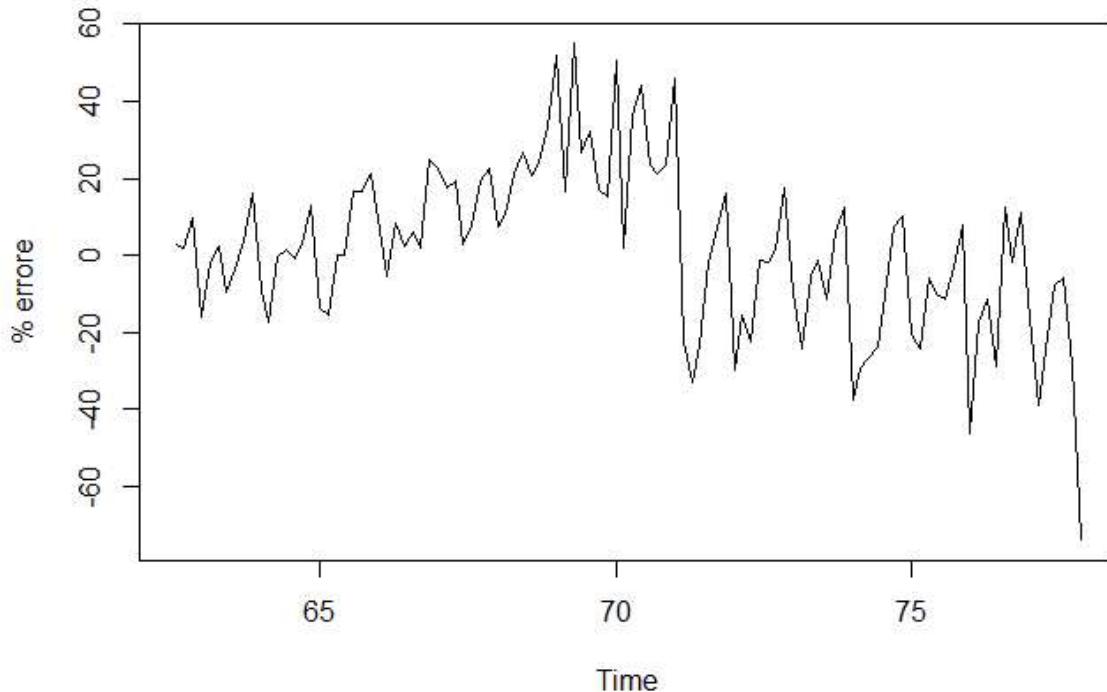
```
## [1] 3321.482
```

```
rmse(test, pred_mod4$mean)
```

```
## [1] 4601.23
```

```
err_plot(test, pred_mod4$mean)
```

Errore percentuale di previsione



Confronto modelli

```
train_eval <- data.frame(Modello=c("Modello1", "Modello2", "Modello3", "Modello4"),
                           RMSE = c(rmse(train, mod1_auto$fitted),
                                     rmse(train, mod2_auto$fitted),
                                     rmse(train, mod3$fitted),
                                     rmse(train, mod4$fitted)),
                           MAE = c(mae(train, mod1_auto$fitted),
                                    mae(train, mod2_auto$fitted),
                                    mae(train, mod3$fitted),
                                    mae(train, mod4$fitted)),
                           MAPE = c(mape(train, mod1_auto$fitted),
                                     mape(train, mod2_auto$fitted),
                                     mape(train, mod3$fitted),
                                     mape(train, mod4$fitted)),
                           AIC = c(mod1_auto$aic, mod2_auto$aic, mod3$aic, mod4$aic))
```

```
test_eval <- data.frame(Modello=c("Modello1", "Modello2", "Modello3", "Modello4"),
                           RMSE = c(rmse(test, pred_mod1$mean),
                                     rmse(test, pred_mod2$mean),
                                     rmse(test, pred_mod3$mean),
                                     rmse(test, pred_mod4$mean)),
```

```

MAE = c(mae(test, pred_mod1$mean),
        mae(test, pred_mod1$mean),
        mae(test, pred_mod3$mean),
        mae(test, pred_mod4$mean)),
MAPE = c(mape(test, pred_mod1$mean),
         mape(test, pred_mod2$mean),
         mape(test, pred_mod3$mean),
         mape(test, pred_mod4$mean)))

```

`train_eval`

Modello	RMSE	MAE	MAPE	AIC
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
Modello1	3056.285	2001.647	12.663471	4705.410
Modello2	3014.676	2060.132	12.927373	4752.916
Modello3	2675.203	1808.782	11.378211	4708.675
Modello4	2155.133	1630.344	9.825467	4492.912

4 rows

`test_eval`

Modello	RMSE	MAE	MAPE
<chr>	<dbl>	<dbl>	<dbl>
Modello1	4851.057	3584.255	17.42545
Modello2	4444.286	3584.255	19.71614
Modello3	3680.758	2767.864	15.72596
Modello4	4601.230	3321.482	16.46479

4 rows

Evaluation: Confronto forecasting performance dei modelli

Scelta misura di *forecasting accuracy*

Al fine di confrontare i diversi modelli definiti precedentemente è importante valutare le performance in termini forecasting accuracy. Prima di definire la procedura di calcolo della forecasting accuracy è importante selezionare la propria misura di accuratezza.

Si sceglie di utilizzare due misure di accuratezza (**MAE, RMSE e MSE**) di tipo **scale-dependent** perchè:

- Misure di accuratezza basate su percentage errors e scaled errors (unit-free) sono indicate per comparare le performance di forecasting tra data set diversi
- Errori di tipo scale-dependent sono nella stessa scala dei dati, ma in questo caso non è un problema
- Si evitano tutte le problematiche relative alle misure di percentage errors legate:
 - al fatto che l'errore viene diviso per il valore dell'osservazione (valori infiniti per y che tende a 0)
 - penalità maggiori per errori negativi piuttosto che quelli positivi

Procedura di *Cross-Validation con rolling forecasting origin*

Effettuo la procedura di *Time-Series Cross Validation* o *Evaluating on a Rolling Forecasting Origin* al fine di:

- Confrontare i diversi modelli in termini di Performance di Forecasting
- Potrò allenare i modelli sull'intero dataset e non perdere i dati del test set (essendo questo già piccolo)
- Ottenere comunque due misure di performance di accuratezza di forecasting accuracy
- Determino la forecasting accuracy (come MAE/RMSE/MSE) per k-step avanti dove k (chiamato h, orizzonte, nella funzione `tsCV()`) sarà il mio orizzonte di previsione nel periodo covid
 - Confrontare l'andamento dei diversi modelli all'aumentare degli step ci consentirà di selezionare
- Parametri di cross-validation -> Scelgo come tipologia di Time Series Cross Validation quella **Constant Holdout**
 - Scelgo arbitrariamente una finestra iniziale (**fixed origin**) di training (il numero minimo di osservazioni necessario a stimare il modello) come 60 -> parametro `initial`
 - **Non-Constant Holdout** -> in modo da utilizzare tutti i dati per il training (altrimenti il training si fermerebbe all'osservazione n-h)
 - **Non-Constant In-Sample** -> Non impostiamo il parametro `window`, che altrimenti andrebbe a settare un dimensione fissata del training set e quindi una moving window

Confronto tra i modelli

Cross-Validation

```
# Calcolo tempo di computazione
start.time <- Sys.time()
print(start.time)
```

```
## [1] "2022-12-09 11:16:58 CET"
```

Definisco la forecast-function per i diversi modelli

```
#1_ARIMA
f_mod1 <- function(x, h, xreg, newxreg) {
  forecast(Arima(x,
    order = c(1, 0, 0),
    seasonal = list(order = c(0, 1, 2)),
    include.constant = FALSE,
    lambda = "auto"),
    h = h)
}
```

```
#2_ARIMA
f_mod2 <- function(x, h, xreg, newxreg) {
  forecast(Arima(x,
    order = c(2, 1, 3),
    xreg = xreg,
    include.constant = TRUE,
    lambda = "auto"),
    h = h,
    xreg=newxreg)
}
```

```
#3_ARIMA
f_mod3 <- function(x, h, xreg, newxreg) {
  forecast(Arima(x,
    order = c(3, 1, 2),
    seasonal = list(order = c(1, 0, 1)),
    xreg = xreg,
    include.constant = TRUE,
    lambda = "auto"),
    h = h,
    xreg=newxreg)
}
```

```
#4_SARIMAX
f_mod4 <- function(x, h, xreg, newxreg) {
```

```

forecast(Arima(x,
  order = c(3, 1, 3),
  seasonal = list(order = c(1, 0, 0)),
  xreg = xreg,
  include.constant = TRUE,
  lambda = "auto"),
  h = h,
  xreg=newxreg)
}

```

```

# Parametri di cross-validation globali per i primi due modelli
h = 42 # 6 settimane
initial = 126 # 3 volte l'orizzonte

# Stima degli errori di previsione con CV

e1 <- tsCV(y = vendite_r1, forecastfunction = f_mod1, h=h, initial = initial)
e1 <- tail(e1, n = -initial)

xreg2 <- dum_day
e2 <- tsCV(y = vendite_r1, forecastfunction = f_mod2, h=h, xreg=xreg2, initial = initial)
e2 <- tail(e2, n = -initial)

end.time <- Sys.time()
print(end.time)

```

```
## [1] "2022-12-09 11:18:11 CET"
```

```

time.taken <- end.time - start.time
print(time.taken)

```

```
## Time difference of 1.215087 mins
```

```
# Parametri di cross-validation globali per il terzo e quarto modello
```

```

h = 42 # 6 settimane
initial = 378 #126*3, modelli maggiormente complessi richiedono un training set iniziale più ampio

```

```
start.time <- Sys.time()
print(start.time)
```

```
## [1] "2022-12-09 11:18:11 CET"
```

```
xreg3 <- four
e3 <- tsCV(y = vendite_r1, forecastfunction = f_mod3, h=h, xreg=xreg3, initial = initial)
e3 <- tail(e3, n = -initial)

xreg4 <- dum_ann
e4 <- tsCV(y = vendite_r1, forecastfunction = f_mod4, h=h, xreg=xreg4, initial = initial)
e4 <- tail(e4, n = -initial)

end.time <- Sys.time()
print(end.time)
```

```
## [1] "2022-12-09 11:44:42 CET"
```

```
time.taken <- end.time - start.time
print(time.taken)
```

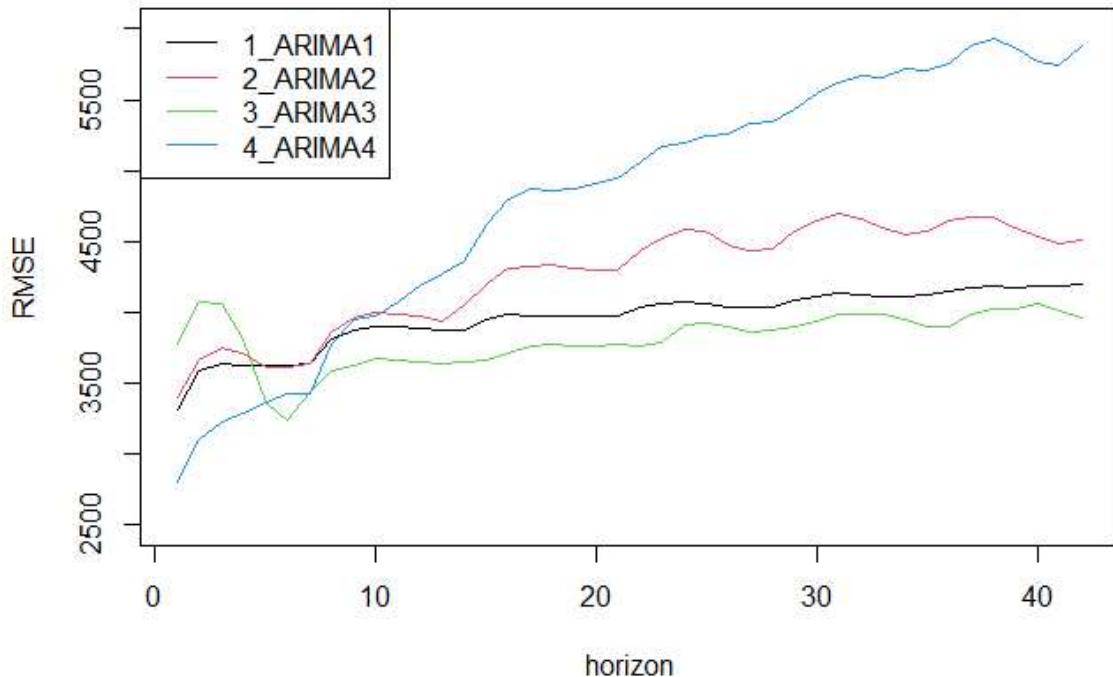
```
## Time difference of 26.52156 mins
```

Accuratezze

```
RMSE_mod1 <- sqrt(colMeans(e1^2, na.rm = TRUE))
RMSE_mod2 <- sqrt(colMeans(e2^2, na.rm = TRUE))
RMSE_mod3 <- sqrt(colMeans(e3^2, na.rm = TRUE))
RMSE_mod4 <- sqrt(colMeans(e4^2, na.rm = TRUE))
#RMSE_mod6 <- sqrt(colMeans(e6^2, na.rm = TRUE))

# Zoom in
plot(1:42, RMSE_mod1, type="l", col=1, xlab="horizon", ylab="RMSE", ylim = c(2500,6000))
lines(1:42, RMSE_mod2, type="l", col=2)
lines(1:42, RMSE_mod3, type="l", col=3)
```

```
lines(1:42, RMSE_mod4, type="l", col=4)
legend("topleft", legend=c("1_ARIMA1", "2_ARIMA2", "3_ARIMA3", "4_ARIMA4"), col=1:4, lty=1)
```



RMSE Medi

```
mean(RMSE_mod1)
```

```
## [1] 3961.703
```

```
mean(RMSE_mod2)
```

```
## [1] 4264.968
```

```
mean(RMSE_mod3)
```

```
## [1] 3812.211
```

```
mean(RMSE_mod4)
```

```
## [1] 4807.045
```