

# ARIMA - Ristorante1

- Setting & Function
- Load data
  - Data preparation
  - Stagionalità
    - Partizionamento del dataset
- Stazionarietà della serie storica
  - Differenza stagionale
- Identificazione del modello
  - Modello1
    - Diagnostica:
    - Modello1 ridotto
    - Previsioni
  - Modello2
    - Diagnostica
    - Modello2 ridotto
    - Previsioni
  - Modello3
    - Diagnostica
    - Modello3 ridotto
    - Previsioni
    - Considerazioni sui primi tre modelli
  - Modello 4 SARIMAX
    - Pre-processing
    - Partizionamento del dataset
    - Stima del modello
      - Diagnostica
      - Previsioni
- Evaluation: Confronto **forecasting performance** dei modelli
  - Scelta misura di *forecasting accuracy*
  - Procedura di *Cross-Validation* con *rolling forecasting origin*
    - Test su Mod1
    - Confronto tra i modelli

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

# Setting & Function

```
set.seed(100)

# Setting librerie utili
# Package names
packages <- c("readxl", "readr", "forecast", "dplyr", "ggplot2",
            "lubridate", "KFAS", "tseries", "xts", "fastDummies")

# Install packages if not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# Packages Loading
invisible(lapply(packages, library, character.only = TRUE))
```

```
## Warning: il pacchetto 'readxl' è stato creato con R versione 4.2.1
```

```
## Warning: il pacchetto 'readr' è stato creato con R versione 4.2.1
```

```
## Warning: il pacchetto 'forecast' è stato creato con R versione 4.2.1
```

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

```
##
## Caricamento pacchetto: 'dplyr'
```

```
## I seguenti oggetti sono mascherati da 'package:stats':  
##  
##     filter, lag
```

```
## I seguenti oggetti sono mascherati da 'package:base':  
##  
##     intersect, setdiff, setequal, union
```

```
## Warning: il pacchetto 'lubridate' è stato creato con R versione 4.2.1
```

```
##  
## Caricamento pacchetto: 'lubridate'
```

```
## I seguenti oggetti sono mascherati da 'package:base':  
##  
##     date, intersect, setdiff, union
```

```
## Warning: il pacchetto 'KFAS' è stato creato con R versione 4.2.1
```

```
## Please cite KFAS in publications by using:  
##  
##     Jouni Helske (2017). KFAS: Exponential Family State Space Models in R. Journal of Statist
```

```
◀ ▶
```

```
## Warning: il pacchetto 'tseries' è stato creato con R versione 4.2.1
```

```
## Warning: il pacchetto 'xts' è stato creato con R versione 4.2.1
```

```
## Caricamento del pacchetto richiesto: zoo
```

```
##  
## Caricamento pacchetto: 'zoo'
```

```
## I seguenti oggetti sono mascherati da 'package:base':  
##  
##     as.Date, as.Date.numeric
```

```
##  
## Caricamento pacchetto: 'xts'
```

```
## I seguenti oggetti sono mascherati da 'package:dplyr':  
##  
##     first, last
```

```
## Warning: il pacchetto 'fastDummies' è stato creato con R versione 4.2.2
```

```
# Setting working directory  
# working_dir = "C:/Users/marco/OneDrive/UNIMIB_DataScience/99-  
#                 PROJECTS/DataScienceLab2022/Dati ristoranti"  
# setwd(working_dir)  
  
# MAPE  
mape <- function(actual,pred){  
  mape <- mean(abs((actual - pred)/actual))*100  
  return (mape)  
}  
  
#MAE  
mae <- function(actual,pred){  
  mae <- mean(abs((actual - pred)))  
  return (mae)  
}  
#MSE  
rmse <- function(actual, pred){  
  rmse <- sqrt(mean((actual - pred)^2))  
  return (rmse)  
}
```

```
# Significatività dei parametri
pars_test <- function(coef, var_coef){
  test <- (1-pnorm(abs(coef)/sqrt(diag(var_coef))))^2
  return(test)
}

# Grafico errore percentuale
err_plot <- function(actual, pred){
  require(xts)
  err_perc <- ((actual - xts(pred, order.by = index(actual)))/(xts(actual, order.by =
    index(actual)))*100
  return(plot(err_perc, ylab="% errore", main="Errore percentuale di previsione"))
}
```

## Load data

```
r1 <- read.csv("../\\Dati ristoranti\\pre-covid_r1.csv")
r1$data <- parse_date(r1$data, "%Y-%m-%d", locale = locale("it"))
head(r1)
```

X...	X	id_ristorante	Location	Regione	Provincia	data	scontrini	
		<int>	<chr>	<chr>	<chr>	<date>	<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

## Data preparation

```
# ts vendite
vendite_r1 <- r1[, c(7,9)]
head(vendite_r1)
```

	data	lordinototale
	<date>	<dbl>
1	2018-09-03	16201.27

	data	lordototale
	<date>	<dbl>
2	2018-09-04	16409.71
3	2018-09-05	15374.10
4	2018-09-06	16531.07
5	2018-09-07	22057.70
6	2018-09-08	28190.46

6 rows

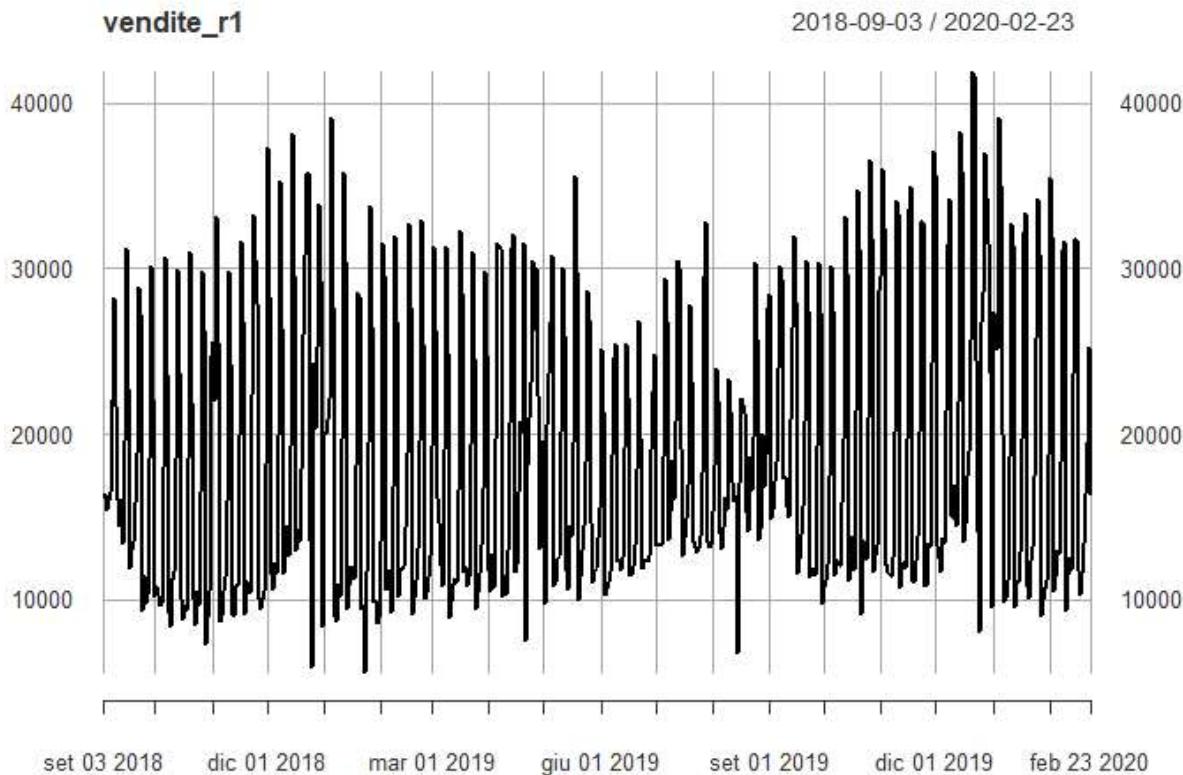
```
str(vendite_r1)
```

```
## 'data.frame':    539 obs. of  2 variables:
##   $ data      : Date, format: "2018-09-03" "2018-09-04" ...
##   $ lordototale: num  16201 16410 15374 16531 22058 ...
```

```
# Trasformo ts in oggetto xts
vendite_r1 <- as.xts(vendite_r1[,-1], order.by = vendite_r1[,1])
head(vendite_r1)
```

```
##           [,1]
## 2018-09-03 16201.27
## 2018-09-04 16409.71
## 2018-09-05 15374.10
## 2018-09-06 16531.07
## 2018-09-07 22057.70
## 2018-09-08 28190.46
```

```
plot(vendite_r1)
```



## Stagionalità

Confrontiamo la media degli incassi dei giorni feriali (non festivi) rispetto alla media del weekend (considero weekend Venerdì-Sabato-Domenica)

```
# Media dei giorni feriali

feriali <- r1[r1$Weekend == 'False', c("data", "lordototale", "Giorno", "Festivo")]
feriali <- feriali[feriali$Festivo == "False",]
tapply(feriali$lordototale, feriali$Giorno, mean)
```

```
##     Monday Thursday Tuesday Wednesday
## 11962.47 13520.99 12033.96 13089.51
```

```
# Media dei giorni "weekend"

weekend <- r1[r1$Weekend == 'True', c("data", "lordototale", "Giorno", "Festivo")]
# weekend <- weekend[weekend$Festivo == "False",]
tapply(weekend$lordototale, weekend$Giorno, mean)
```

```
##   Friday Saturday Sunday
## 19922.83 31510.69 27733.93
```

```
# Confronto media giorni Lun-gio in base a se è festivo o meno
festivo_noweekend <- r1[r1$Weekend == 'False',c("data", "lordototale", "Giorno",
      "Festivo")]
tapply(festivo_noweekend$lordototale, festivo_noweekend$Festivo, mean)
```

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

```
# Costruzione dummy per modellare stagionalità a 7 giorni
```

```
require('fastDummies')
dum_day <- r1[, c("data", "Giorno")]
dum_day[, "Giorno"] <- as.factor(dum_day$Giorno)
dum_day <- dummy_cols(dum_day, select_columns = c("Giorno"),
      remove_first_dummy = TRUE,
      remove_selected_columns = TRUE)
dum_day
```

data	Giorno_Friday	Giorno_Monday	Giorno_Saturday	Giorno_Sunday
<date>	<int>	<int>	<int>	<int>
2018-09-03	0	1	0	0
2018-09-04	0	0	0	0
2018-09-05	0	0	0	0
2018-09-06	0	0	0	0
2018-09-07	1	0	0	0
2018-09-08	0	0	1	0
2018-09-09	0	0	0	1
2018-09-10	0	1	0	0
2018-09-11	0	0	0	0
2018-09-12	0	0	0	0

1-10 of 539 rows | 1-5 of 8 columns

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

```
dum_day_rid <- subset(dum_day, select = -Giorno_Monday)
dum_day_rid
```

	<b>data</b>	<b>Giorno_Friday</b>	<b>Giorno_Saturday</b>	<b>Giorno_Sunday</b>	<b>Giorno_Thursday</b>
	<date>	<int>	<int>	<int>	<int>
1	2018-09-03	0	0	0	0
2	2018-09-04	0	0	0	0
3	2018-09-05	0	0	0	0
4	2018-09-06	0	0	0	1
5	2018-09-07	1	0	0	0
6	2018-09-08	0	1	0	0
7	2018-09-09	0	0	1	0
8	2018-09-10	0	0	0	0
9	2018-09-11	0	0	0	0
10	2018-09-12	0	0	0	0

1-10 of 539 rows | 1-6 of 8 columns      Previous [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) ... [54](#) [Next](#)

```
# __IGNORE__ per i primi tre modelli. Userò solo le dummy a 7 giorni per la modellazione della stagionalità
```

```
# Costruzione dummuy per weekend e festivi
# dum_week <- r1[, c("data", "Festivo")]
# dum_week[, "Festivo"] <- as.factor(dum_week$Festivo)
# dum_week <- dummy_cols(dum_week, select_columns = "Festivo",
#                         remove_first_dummy = TRUE,
#                         remove_selected_columns = TRUE)
# dum_week
```

```
# xts object con tutte le variabili dummy
# dum <- cbind(dum_day, dum_week)
# dum <- xts(dum[, -1], dum$data)
# dum <- subset(dum, select = -c(data))
# head(dum)
```

```
dum_day_xts <- xts(dum_day[, -1], dum_day$data)
head(dum_day_xts)
```

	<b>Giorno_Friday</b>	<b>Giorno_Monday</b>	<b>Giorno_Saturday</b>	<b>Giorno_Sunday</b>
## 2018-09-03	0	1	0	0
## 2018-09-04	0	0	0	0
## 2018-09-05	0	0	0	0

```
## 2018-09-06      0      0      0      0
## 2018-09-07      1      0      0      0
## 2018-09-08      0      0      1      0
##           Giorno_Thursday Giorno_Tuesday Giorno_Wednesday
## 2018-09-03      0      0      0
## 2018-09-04      0      1      0
## 2018-09-05      0      0      1
## 2018-09-06      1      0      0
## 2018-09-07      0      0      0
## 2018-09-08      0      0      0
```

```
dum_day_xts_rid <- xts(dum_day_rid[, -1], dum_day$data)
head(dum_day_xts_rid)
```

```
##           Giorno_Friday Giorno_Saturday Giorno_Sunday Giorno_Thursday
## 2018-09-03      0      0      0      0
## 2018-09-04      0      0      0      0
## 2018-09-05      0      0      0      0
## 2018-09-06      0      0      0      1
## 2018-09-07      1      0      0      0
## 2018-09-08      0      1      0      0
##           Giorno_Tuesday Giorno_Wednesday
## 2018-09-03      0      0
## 2018-09-04      1      0
## 2018-09-05      0      1
## 2018-09-06      0      0
## 2018-09-07      0      0
## 2018-09-08      0      0
```

```
# df <- cbind(vendite_r1, dum_day_xts)
# head(df)
```

```
df <- cbind(vendite_r1, dum_day_xts_rid)
head(df)
```

```
##           vendite_r1 Giorno_Friday Giorno_Saturday Giorno_Sunday
## 2018-09-03  16201.27      0      0      0
## 2018-09-04  16409.71      0      0      0
## 2018-09-05  15374.10      0      0      0
```

```

## 2018-09-06 16531.07 0 0 0
## 2018-09-07 22057.70 1 0 0
## 2018-09-08 28190.46 0 1 0
## Giorno_Thursday Giorno_Tuesday Giorno_Wednesday
## 2018-09-03 0 0 0
## 2018-09-04 0 1 0
## 2018-09-05 0 0 1
## 2018-09-06 1 0 0
## 2018-09-07 0 0 0
## 2018-09-08 0 0 0

```

```
write.zoo(df, file="..\Dati aggiuntivi\Dat CV_Arima\df_123.csv", sep=",")
```

## Partizionamento del dataset

```

# Divisione training-test set
train_date <- nrow(df) *0.8
train_temp <- df[1:train_date,]
test <- df[-c(1:train_date),] # Usare alla fine

```

```

# Training - validation set
train_date_rid <- nrow(train_temp)*0.9
train <- train_temp[1:train_date_rid,]
validation <- train_temp[-c(1:train_date_rid),]
rm(list="train_temp")

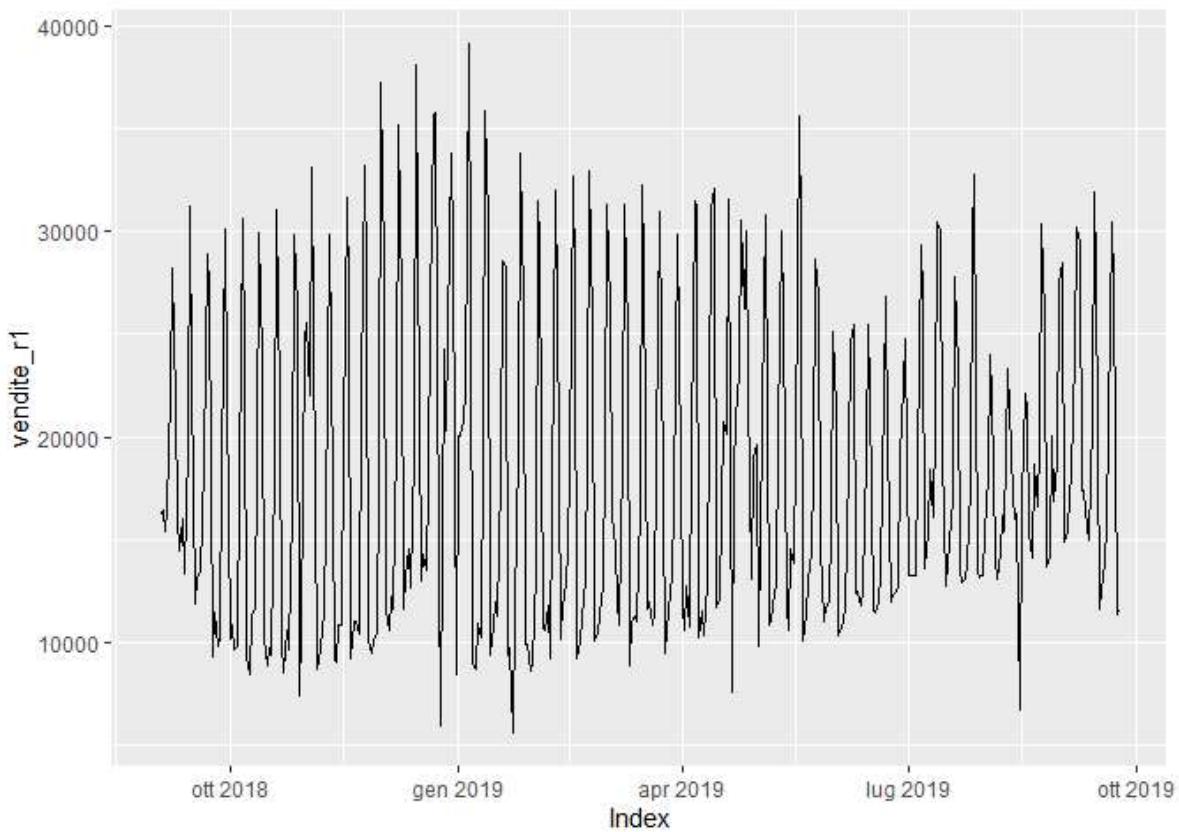
```

## Stazionarietà della serie storica

```

# Intera serie storica
autoplots(train[,1])

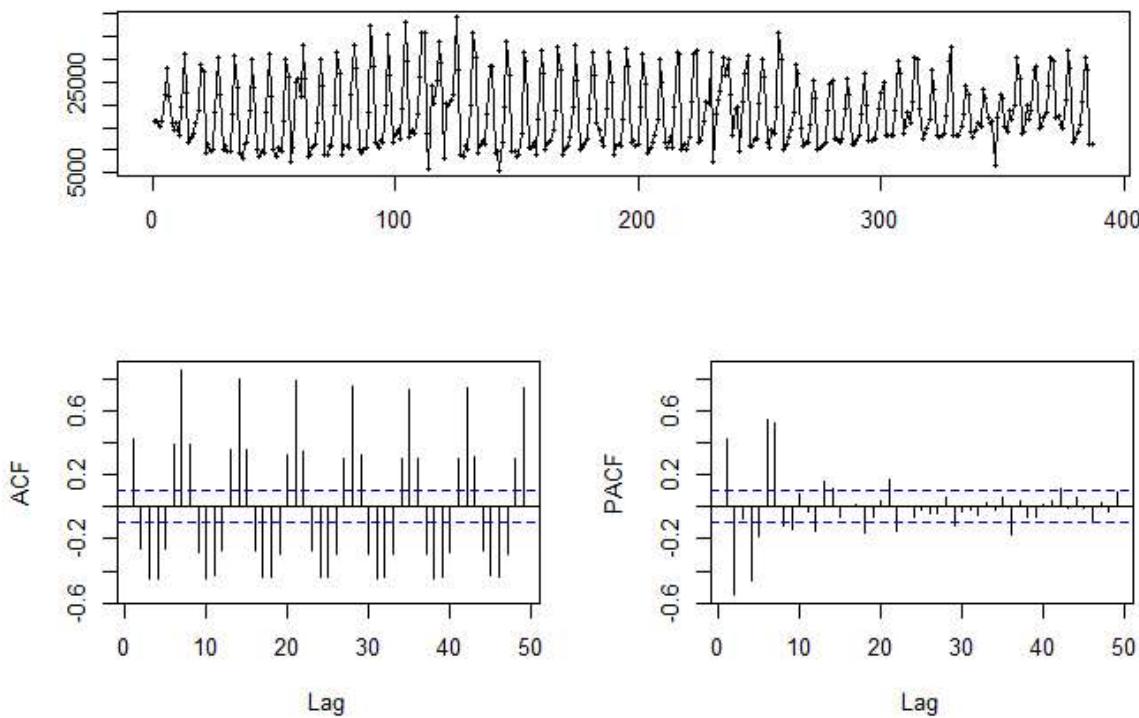
```



La serie storica sembra non avere una tendenza di medio/lungo periodo a crescere o scendere (trend). Valutiamo l'andamento della funzione di autocorrelazione (totale e parziale)

```
tsdisplay(train[,1], lag.max = 49)
```

train[, 1]



ACF risulta avere ritardi significativi che **decadono a zero molto lentamente**, sintomo di una non stazionarietà della serie storica. Notiamo come i ritardi maggiormente significativi sono quelli ai **ritardi stagionali** multipli di 7.

```
#Ljung-Box test
# (a non-stationary signal will have a low p-value)

lag.length = 25
Box.test(train[,1], lag=lag.length, type="Ljung-Box")
```

```
##
## Box-Ljung test
##
## data: train[, 1]
## X-squared = 2038.3, df = 25, p-value < 2.2e-16
```

Il test di Ljung-Box conferma l'**ipotesi di non stazionarietà**.

La non stazionarietà potrebbe però essere dovuta alla **presenza di stagionalità** nella serie storica:

```
# Augmented Dickey-Fuller (ADF) t-statistic test for unit root
options(warn=-1)
```

```
require(tseries)

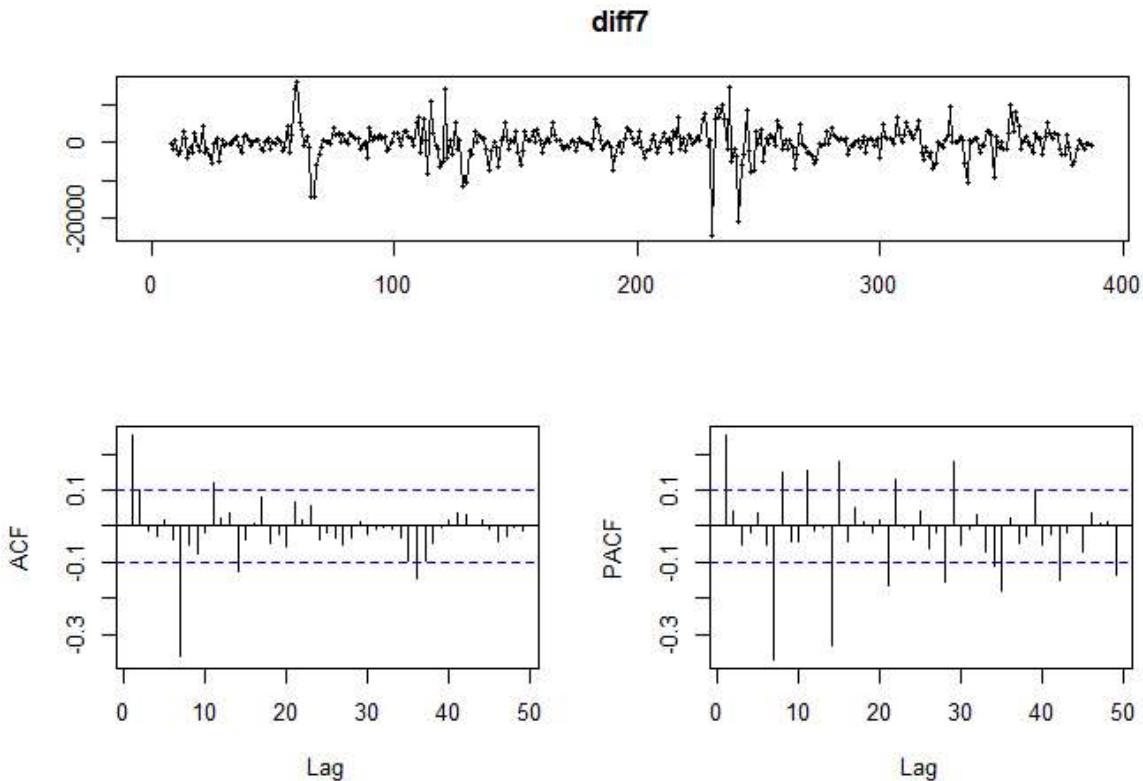
adf.test(train[,1])
```

```
## 
## Augmented Dickey-Fuller Test
##
## data: train[, 1]
## Dickey-Fuller = -3.8715, Lag order = 7, p-value = 0.01565
## alternative hypothesis: stationary
```

Come effettivamente viene suggerito dal **test di Dickey Fuller** (valuta la presenza di radici unitarie)

## Differenza stagionale

```
diff7 <- diff(train[,1], 7)
tsdisplay(diff7, lag.max = 49)
```



```
lag.length = 25
Box.test(diff7, lag=lag.length, type="Ljung-Box")
```

```
##  
## Box-Ljung test  
##  
## data: diff7  
## X-squared = 104.23, df = 25, p-value = 1.205e-11
```

```
rm(list="diff7")
```

Il test indica che **è ancora presente correlazione seriale**. Notiamo però come la ACF, nei primi 25 ritardi, abbia solamente il settimo ritardo significativo, mentre ora la PACF tende a zero molto più lentamente rispetto a prima (con ritardi particolarmente significativi ai ritardi stagionali multipli di 7).

Tutto questo probabilmente è dovuto al fatto che nella serie storica è presente una stagionalità **multipla**, nello specifico una stagionalità a 7 giorni e una stagionalità a 365 giorni.

## Identificazione del modello

In questa fase cercheremo di individuare un modello ARIMA adatto a modellare i nostri dati. Nei primi 3 modelli verrà modellata la stagionalità a 7 giorni attraverso variabili dummy, mentre nel 4 modello introdurremmo anche variabili che modellino una stagionalità a 365 giorni.

### Modello1

- Componente AR stagionale di ordine 1:
- Componente MA stagionale di ordine 1:
- Componente MA non stagionale di ordine 2
- Dummy stagionali come regressori esterni
- Costante inclusa

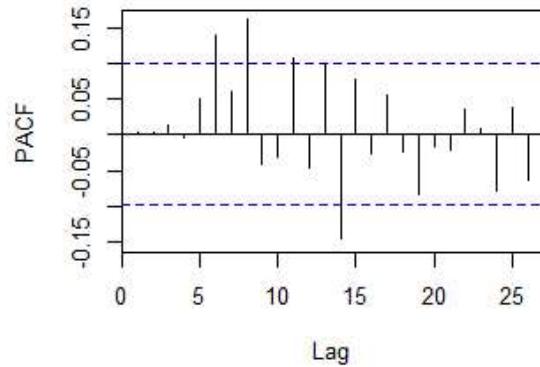
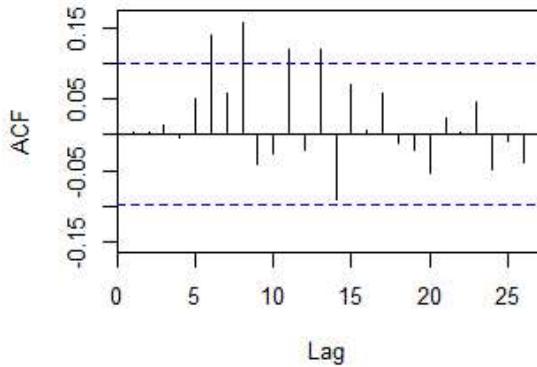
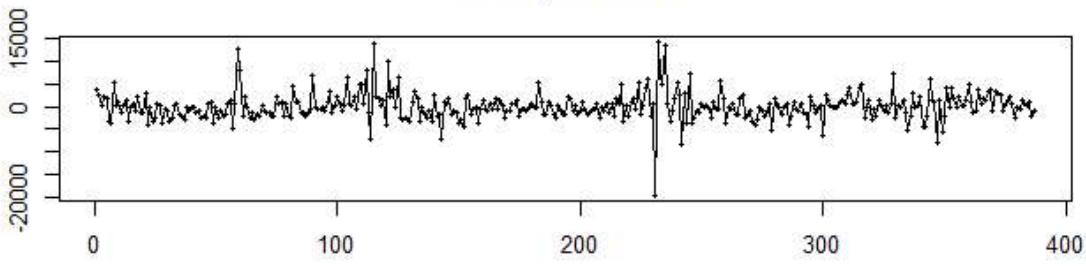
```
mod1 <- Arima(y = train$vendite_r1,  
                 order = c(0, 0, 2),  
                 list(order = c(1, 0, 1), period = 7),  
                 xreg = train[, -1],  
                 include.constant = TRUE,  
                 )
```

### Diagnostica:

```
summary(mod1)
```

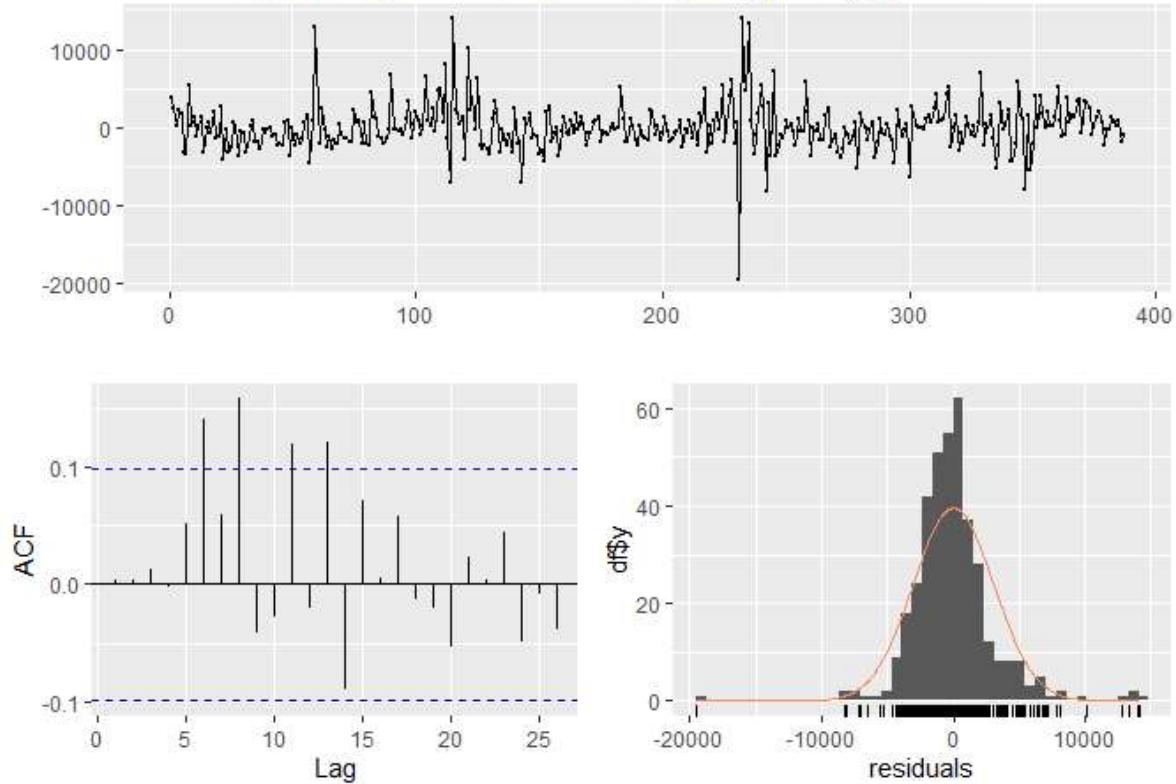
```
## Series: train$vendite_r1
## Regression with ARIMA(0,0,2)(1,0,1)[7] errors
##
## Coefficients:
##             ma1      ma2     sar1     sma1   intercept Giorno_Friday
##             0.3940  0.1735  0.8209 -0.6978  12003.9834      7377.982
## s.e.    0.0532  0.0521  0.1266  0.1640    723.0373     1023.412
##             Giorno_Saturday Giorno_Sunday Giorno_Thursday Giorno_Tuesday
##             18441.4679    14036.8838     1753.004      265.3681
## s.e.      955.1846     806.8891     1022.420      794.7522
##             Giorno_Wednesday
##             1623.5540
## s.e.      944.0975
##
## sigma^2 = 9556949: log likelihood = -3654.03
## AIC=7332.05  AICc=7332.89  BIC=7379.55
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set -6.841803 3047.18 2049.898 -3.93605 13.41876 0.3425153 0.004242026
```

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

**mod1\$residuals**

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

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



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

```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(0,0,2)(1,0,1)[7] errors  
## Q* = 43.472, df = 21, p-value = 0.002735  
##  
## Model df: 4. Total lags used: 25
```

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

```
##          ma1          ma2          sar1          sma1  
## 1.287859e-13 8.764276e-04 8.779399e-11 2.093390e-05  
## intercept Giorno_Friday Giorno_Saturday Giorno_Sunday  
## 0.000000e+00 5.628831e-13 0.000000e+00 0.000000e+00  
## Giorno_Thursday Giorno_Tuesday Giorno_Wednesday  
## 8.642513e-02 7.384547e-01 8.548901e-02
```

I parametri che **risultano essere significativi**, ad un livello  $\alpha = 0.05$ , sono:

- MA(1):  $p\text{-value}: 0.000$
- MA(2):  $p\text{-value}: 0.000$
- SAR(1)[7]:  $p\text{-value}: 0.000$
- SMA(1)[7]:  $p\text{-value}: 0.000$
- Inercetta:  $p\text{-value}: 0.000$
- Giorno\_Saturday  $p\text{-value}: 0.001$
- Giorno\_Sunday  $p\text{-value}: 0.002$
- Giorno\_Friday:  $p\text{-value}: 0.000$

Mentre quelle che **non risultano essere significativi**:

- Giorno\_Thursday  $p\text{-value}: 0.086$
- Giorno\_Tuesday  $p\text{-value}: 0.73$

- Giorno\_Wednesday *p-value*: 0.085

Risultano essere i giorni della settimana **non-weekend**, ovvero tutti i giorni che hanno una **media simile** alla variabile dummy esclusa (Monday)

L'**AIC** risulta essere pari a : 7332.05

L'errore sul training set **MAE** risulta pari a 2049.89 mentre l'**RMSE** pari a: 3047.18

Il **Box-Ljung test** (*p-value* = 0.002735 ) ci indica che i residui presentano ancora auto-correlazione seriale.

## Modello1 ridotto

Ristimo il modello togliendo tutte le variabili non significative:

```
mod1_rid <- Arima(y = train$vendite_r1,
                     order = c(0, 0, 2),
                     list(order = c(1, 0, 1), period = 7),
                     xreg = train[, -c(1,5,6,7)],
                     include.constant = TRUE,
                     )
```

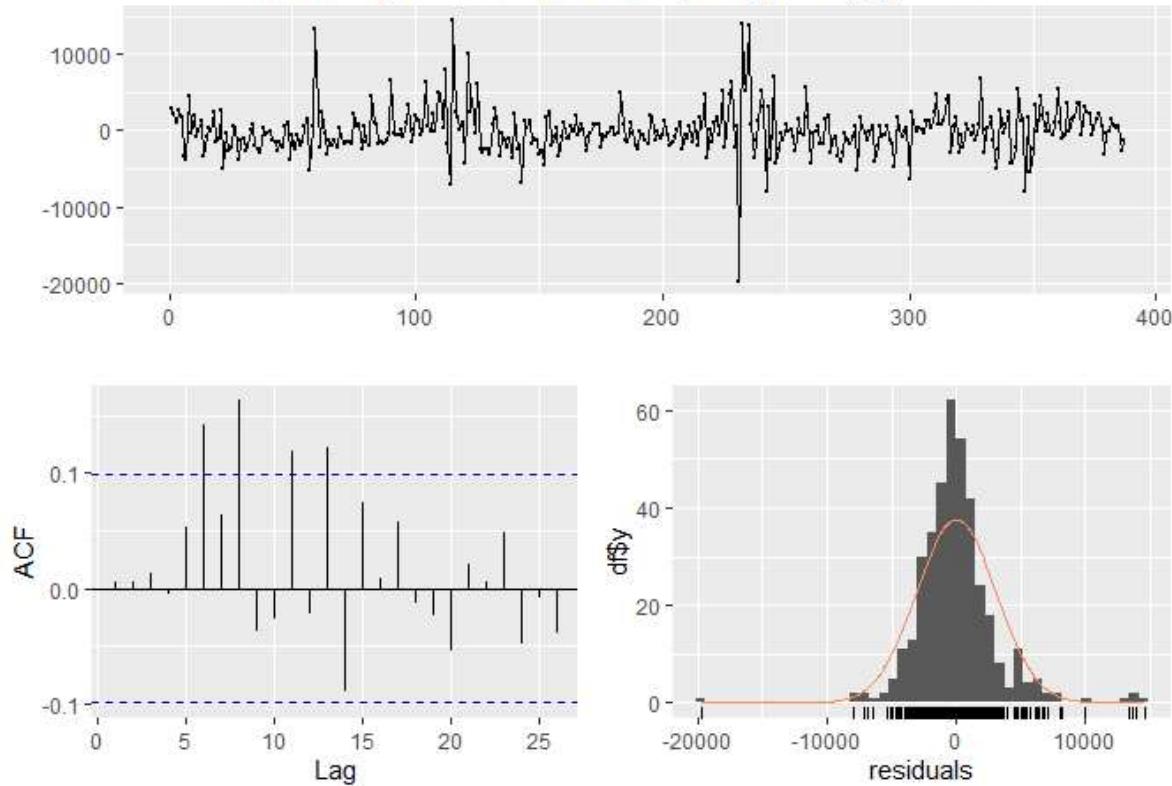
```
summary(mod1_rid)
```

```
## Series: train$vendite_r1
## Regression with ARIMA(0,0,2)(1,0,1)[7] errors
##
## Coefficients:
##             ma1      ma2     sar1     sma1   intercept Giorno_Friday
##             0.3993  0.1785  0.8745 -0.7501  12940.397      6082.2285
## s.e.    0.0516  0.0517  0.0715  0.1002    544.446      843.7682
##             Giorno_Saturday Giorno_Sunday
##             17474.5458    13466.6044
## s.e.        924.4464     849.7595
##
## sigma^2 = 9566103: log likelihood = -3655.94
## AIC=7329.88  AICc=7330.36  BIC=7365.5
##
## Training set error measures:
##                  ME      RMSE       MAE       MPE       MAPE       MASE
## Training set -3.760969 3060.777 2046.821 -3.834681 13.37718 0.3420012
```

```
##          ACF1
## Training set 0.005196787
```

```
checkresiduals(mod1_rid, test = FALSE)
```

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



```
checkresiduals(mod1_rid, test = "LB", lag = 25, plot = FALSE)
```

```
##
## Ljung-Box test
##
## data: Residuals from Regression with ARIMA(0,0,2)(1,0,1)[7] errors
## Q* = 44.672, df = 21, p-value = 0.001912
##
## Model df: 4. Total lags used: 25
```

```
pars_test(mod1_rid$coef, mod1_rid$var.coef)
```

```

##          ma1          ma2         sar1        sma1      intercept
## 1.043610e-14 5.475345e-04 0.000000e+00 7.127632e-14 0.000000e+00
## Giorno_Friday Giorno_Saturday Giorno_Sunday
## 5.659917e-13 0.000000e+00 0.000000e+00

```

Le variabili considerate ora risultano essere tutte significative.

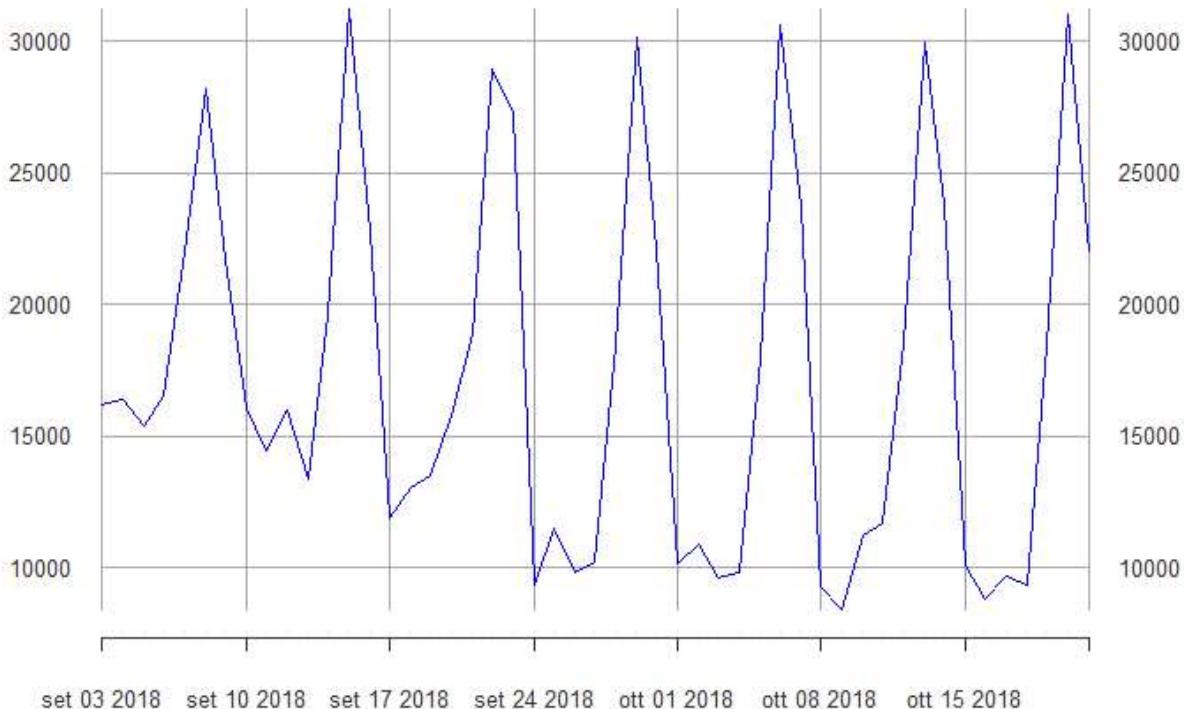
```

# Fit prime 7 settimane
plot(train[1:49,1],
      col = "blue", lwd=0.5,
      main = "Fitted Values")

```

Fitted Values

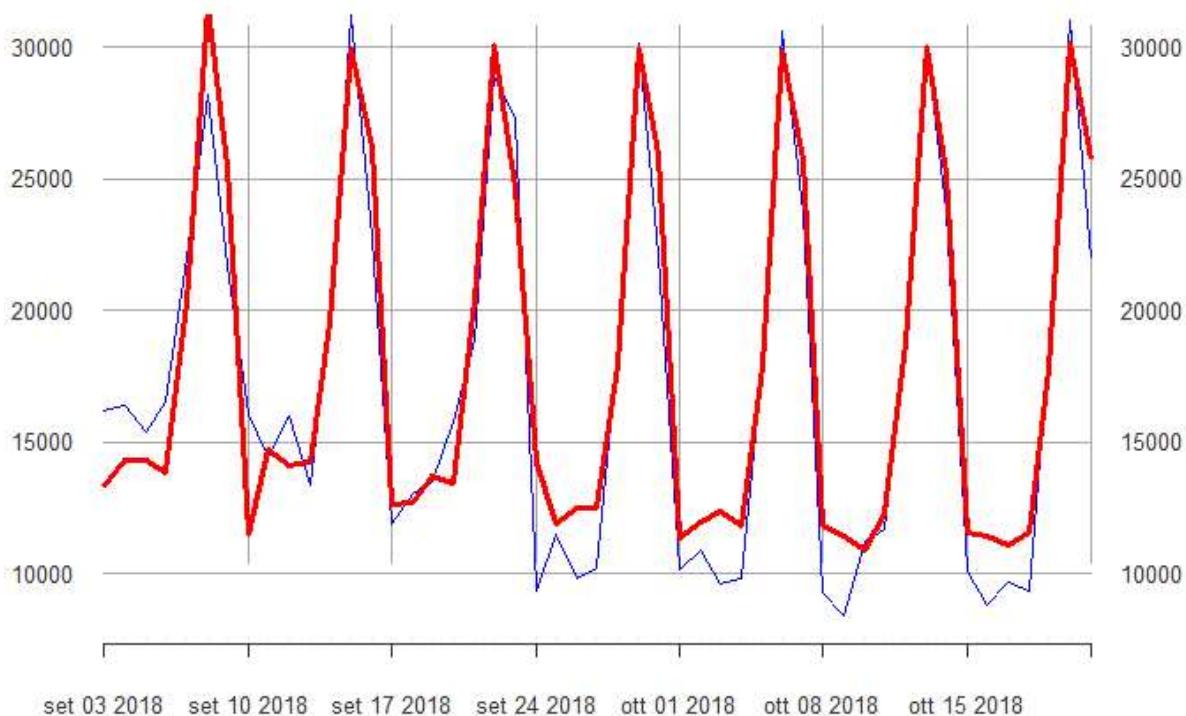
2018-09-03 / 2018-10-21



```
lines(xts(mod1_rid$fitted[1:49], order.by = index(train[1:49])), col="red", lwd=3)
```

**Fitted Values**

2018-09-03 / 2018-10-21



set 03 2018 set 10 2018 set 17 2018 set 24 2018 ott 01 2018 ott 08 2018 ott 15 2018

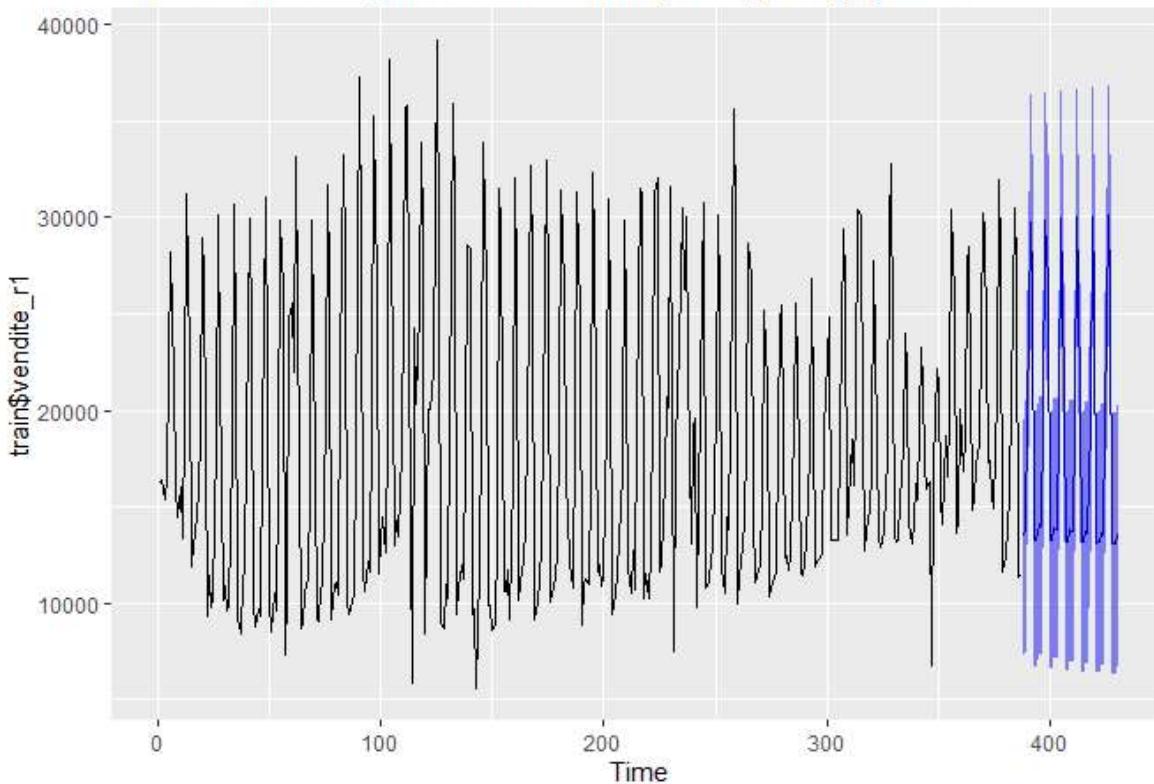
Notiamo come i

picchi stagionali a 7 giorni vengono modellati bene, mentre gli altri comportamenti non vengono ben interpretati dal modello

**Previsioni**

```
# Al momento vengono fatte previsioni 44 passi in avanti (Lunghezza del validation set)
pred_mod1 <- forecast(mod1_rid, h = 44,
                      level = 95,
                      xreg = validation[, -c(1,5,6,7)])
```

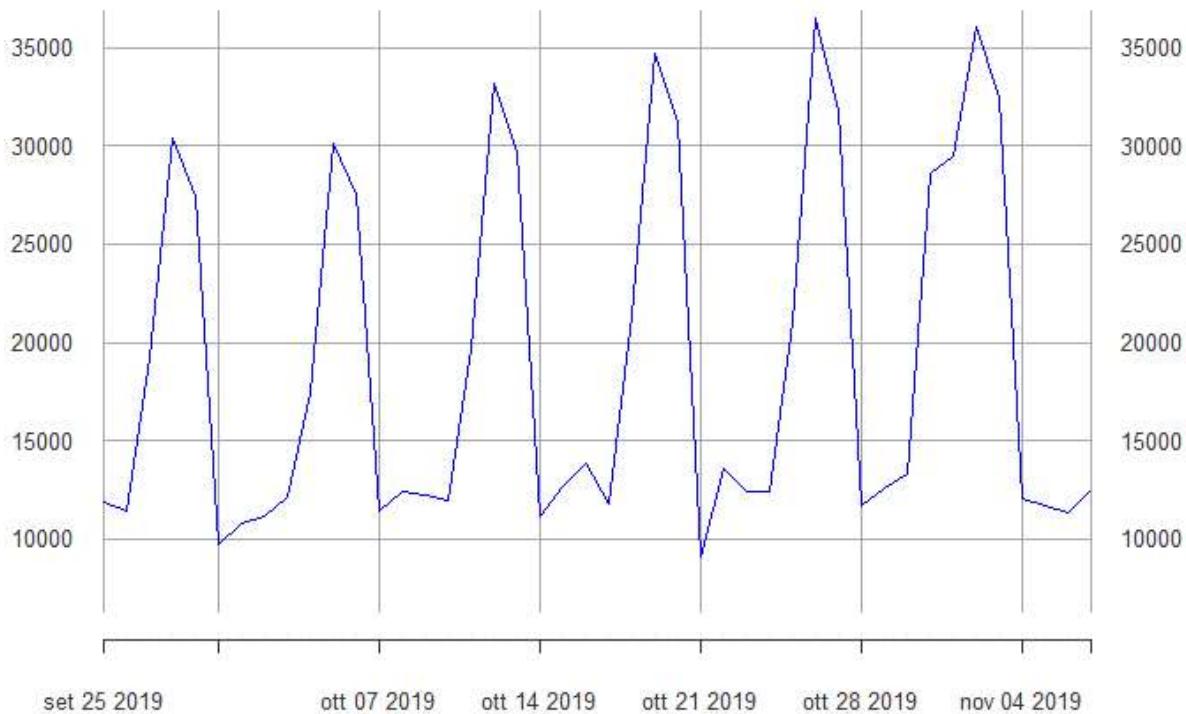
```
autoplot(pred_mod1)
```

**Forecasts from Regression with ARIMA(0,0,2)(1,0,1)[7] errors**

```
plot(validation[,1],  
      col = "blue", lwd=0.5,  
      ylim = c(min(pred_mod1$lower), max(pred_mod1$upper)), main = "Predictions")
```

**Predictions**

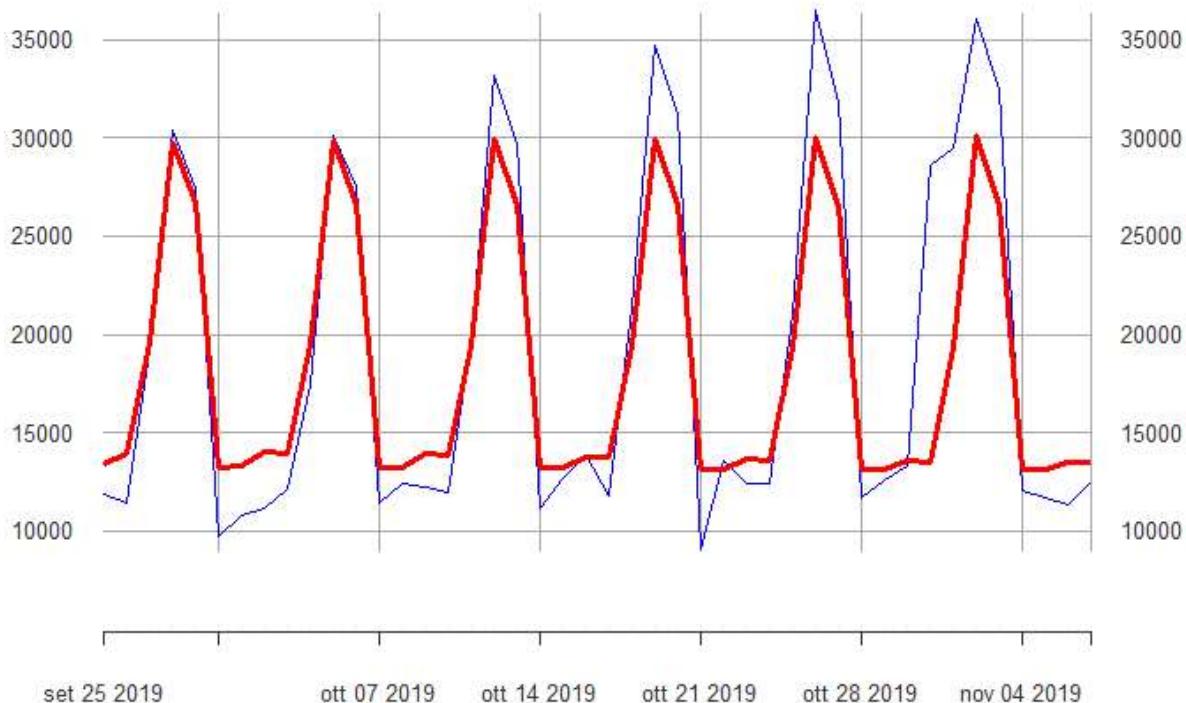
2019-09-25 / 2019-11-07



```
lines(xts(pred_mod1$mean, order.by = index(validation)), col="red", lwd=3)
```

**Predictions**

2019-09-25 / 2019-11-07



```
mape(validation[,1], xts(pred_mod1$mean, order.by = index(validation)))
```

```
## [1] 13.77047
```

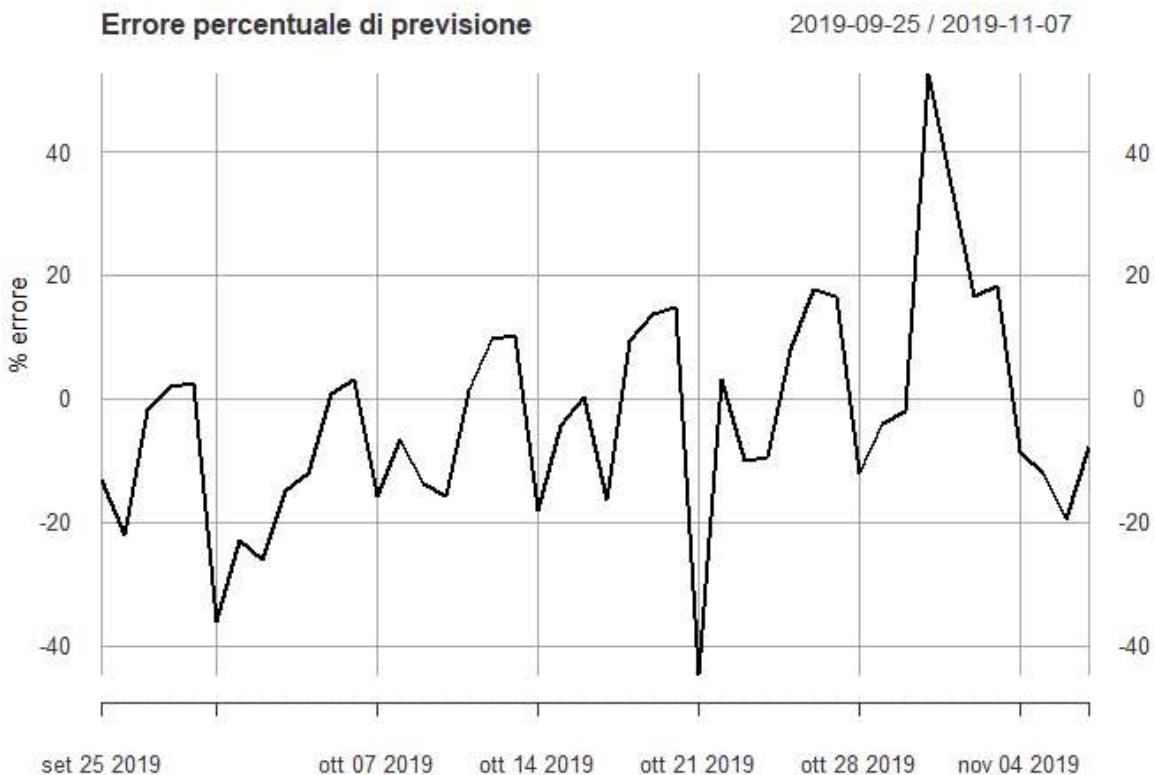
```
mae(validation[,1], xts(pred_mod1$mean, order.by = index(validation)))
```

```
## [1] 2585.921
```

```
rmse(validation[,1], xts(pred_mod1$mean, order.by = index(validation)))
```

```
## [1] 3805.724
```

```
err_plot(validation[,1], xts(pred_mod1$mean, order.by = index(validation)))
```



Sono state effettuate 44 previsioni con il modello considerato e sono state confrontate con il validation set.

Il MAE risulta pari a 2585.921, mentre l'RMSE risulta pari 3805.724. Se confrontiamo tali risultati con gli errori riscontrati sul training set, notiamo come è stato evitato il fenomeno di overfitting, anche se gli errori di previsioni (sia sul training che sul validation) rimangono comunque notevoli.

## Modello2

- 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 5**
- Dummy stagionali
- Costante non inclusa

```
mod2 <- Arima(y = train$vendite_r1,
                 order = c(5, 0, 2),
                 list(order = c(1, 0, 1), period = 7),
                 xreg = train[, -1],
                 include.constant = TRUE,
                 )
```

## Diagnostica

```
summary(mod2)
```

```
## Series: train$vendite_r1
## Regression with ARIMA(5,0,2)(1,0,1)[7] errors
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1      ma2      sar1      sma1
##         1.4745  -0.7976  0.0820  0.0673  0.0655  -1.0969  0.3849  0.8897  -0.8255
## s.e.  0.2580   0.3159  0.1312  0.1034  0.0772   0.2546  0.2115  0.0584   0.0696
##          intercept Giorno_Friday Giorno_Saturday Giorno_Sunday
##            12044.5068      7354.134       18406.7864     14014.0891
## s.e.    870.0329      1121.061       998.0947     779.2012
##          Giorno_Thursday Giorno_Tuesday Giorno_Wednesday
```

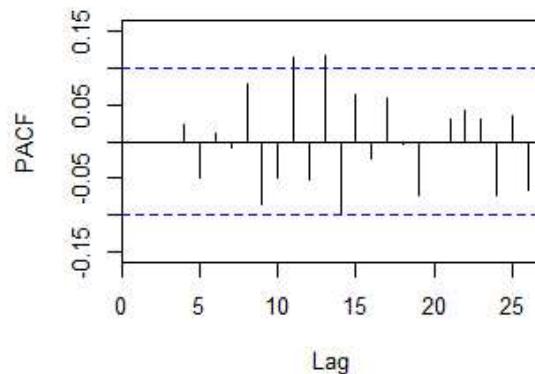
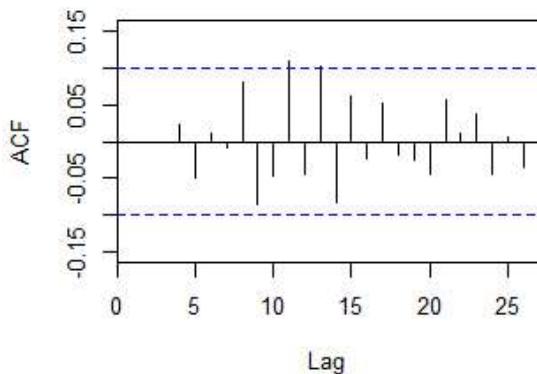
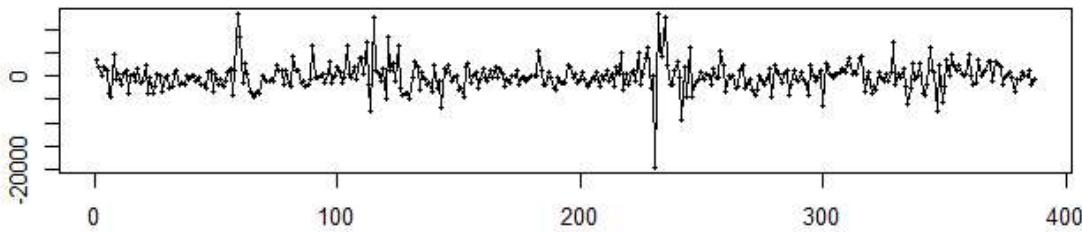
```

##          1740.960      262.3081      1639.9980
## s.e.     1120.702      769.2994      990.3393
##
## sigma^2 = 9264549:  log likelihood = -3645.56
## AIC=7325.11    AICc=7326.77    BIC=7392.41
##
## Training set error measures:
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -18.24604 2980.188 2011.298 -3.757889 13.10979 0.3360657
##           ACF1
## Training set -0.001262591

```

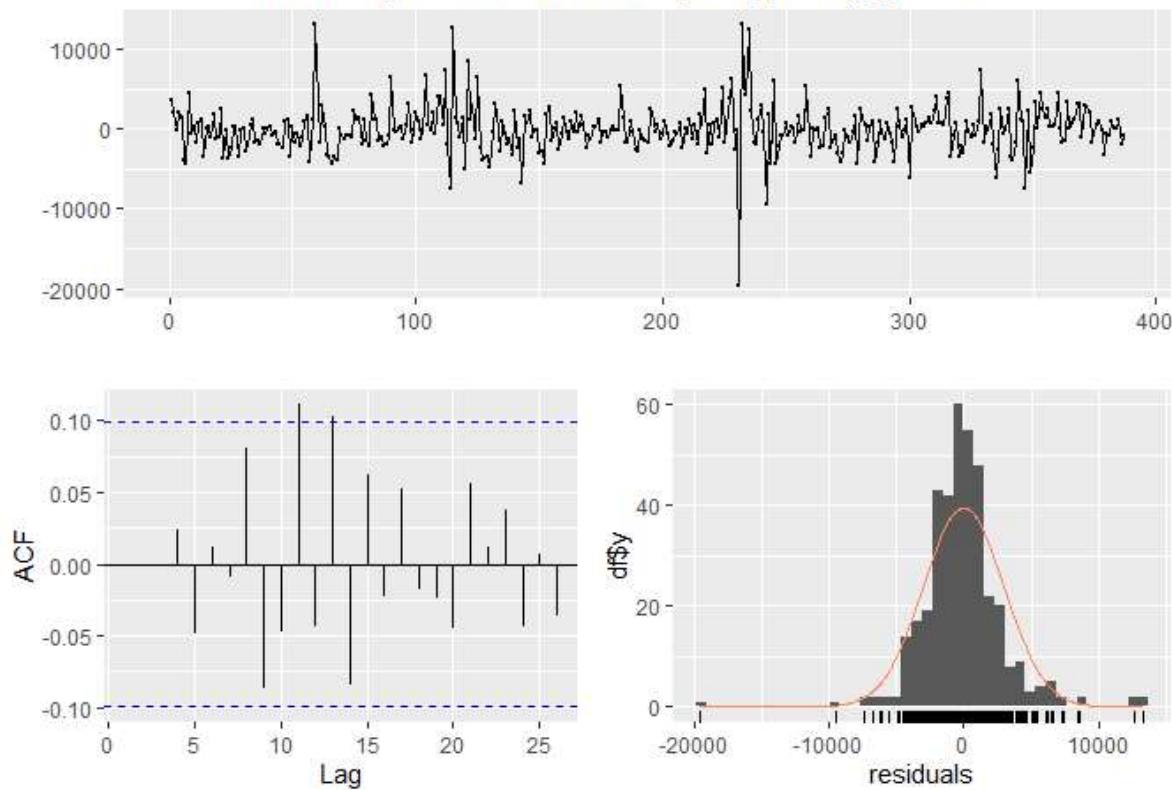
```
tsdisplay(mod2$residuals)
```

**mod2\$residuals**



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

### Residuals from Regression with ARIMA(5,0,2)(1,0,1)[7] errors



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

```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(5,0,2)(1,0,1)[7] errors  
## Q* = 27.348, df = 16, p-value = 0.03777  
##  
## Model df: 9. Total lags used: 25
```

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

	ar1	ar2	ar3	ar4
##	1.095987e-08	1.156824e-02	5.320491e-01	5.151031e-01
##	ar5	ma1	ma2	sar1
##	3.960649e-01	1.643243e-05	6.883863e-02	0.000000e+00
##	sma1	intercept	Giorno_Friday	Giorno_Saturday
##	0.000000e+00	0.000000e+00	5.381606e-11	0.000000e+00

```
## Giorno_Sunday Giorno_Thursday Giorno_Tuesday Giorno_Wednesday
## 0.000000e+00 1.203144e-01 7.331261e-01 9.772261e-02
```

I parametri che **risultano essere significativi**, ad un livello  $\alpha = 0.05$ , sono:

- AR(1):  $p\text{-value}$ : 0.0000
- AR(2):  $p\text{-value}$ : 0.011
- MA(1):  $p\text{-value}$ : 0.000
- SAR(1)[7]:  $p\text{-value}$ : 0.000
- SMA(1)[7]:  $p\text{-value}$ : 0.000
- Intercetta:  $p\text{-value}$ : 0.000
- Giorno\_Saturday  $p\text{-value}$ : 0.000
- Giorno\_Sunday  $p\text{-value}$ : 0.000
- Giorno\_Friday  $p\text{-value}$ : 0.000

Mentre quelle che **non risultano essere significativi**:

- AR(3):  $p\text{-value}$ : 0.532
- AR(4):  $p\text{-value}$ : 0.515
- AR(5):  $p\text{-value}$ : 0.396
- MA(2):  $p\text{-value}$ : 0.068
- Giorno\_Thursday  $p\text{-value}$ : 0.120
- Giorno\_Tuesday  $p\text{-value}$ : 0.733
- Giorno\_Wednesday  $p\text{-value}$ : 0.097

**L'AIC** risulta essere pari a : 7325.11

L'errore sul training set **MAE** risulta pari a 2011.298 mentre l'**RMSE** pari a: 2980.188

Il **Box-Ljung test** ( $p\text{-value} = 0.0377$ ) ci indica che i residui presentano ancora auto-correlazione seriale.

## Modello2 ridotto

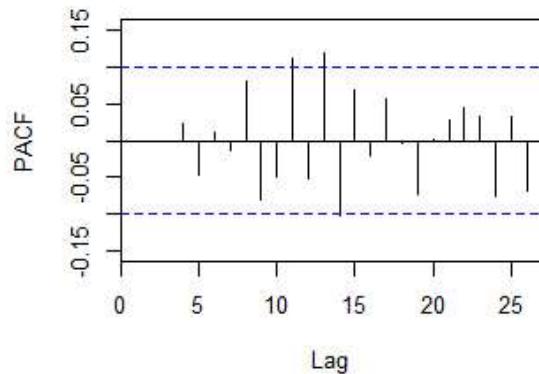
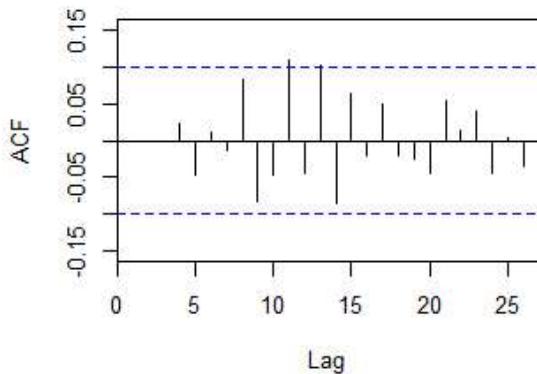
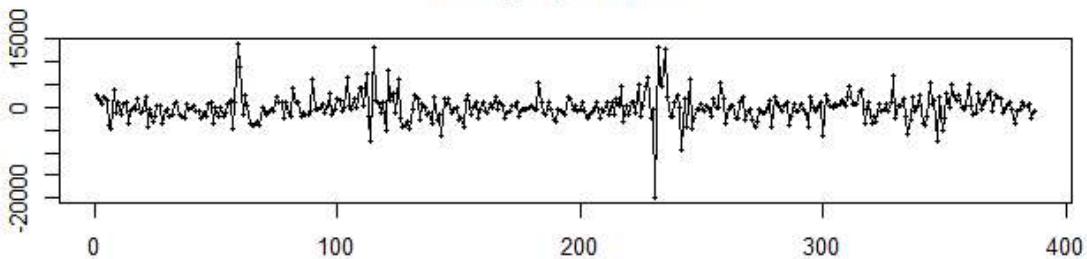
Ristimiamo il modello togliendo le i regressori **stagionali** non significativi, mentre i parametri AR e MA che non sono risultati significativi verranno tolti nel *modello3*:

```
mod2_rid <- Arima(y = train$vendite_r1,
                     order = c(5, 0, 2),
                     list(order = c(1, 0, 1), period = 7),
                     xreg = train[, -c(1,5,6,7)],
                     include.constant = TRUE,
                     )
```

```
summary(mod2_rid)
```

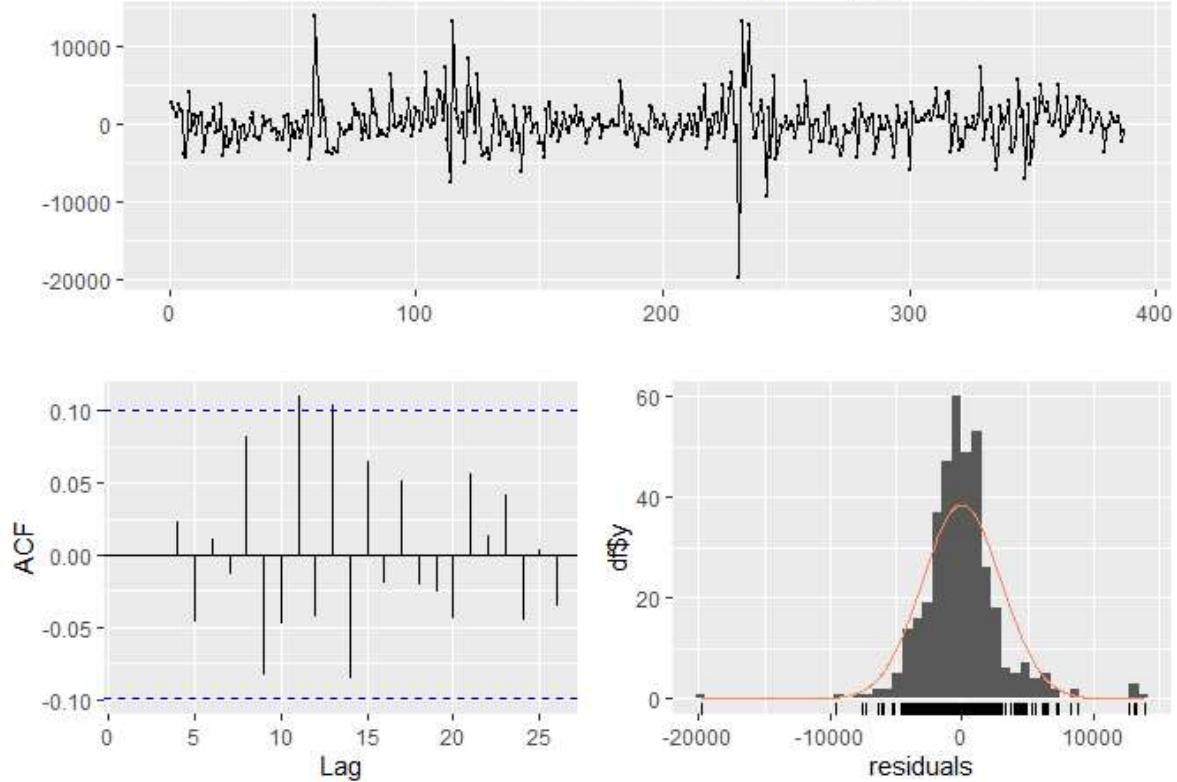
```
## Series: train$vendite_r1
## Regression with ARIMA(5,0,2)(1,0,1)[7] errors
##
## Coefficients:
##             ar1      ar2      ar3      ar4      ar5      ma1      ma2      sar1      sma1
##             1.4786   -0.8058   0.0787   0.0717   0.0658   -1.0987   0.3916   0.9133   -0.8396
## s.e.    0.2512    0.3096   0.1305   0.1042   0.0774    0.2476   0.2074   0.0458    0.0587
##             intercept Giorno_Friday Giorno_Saturday Giorno_Sunday
##             12961.1506      5946.4618      17488.3462      13625.3281
## s.e.    741.4416       832.3216       939.9731       841.7965
##
## sigma^2 = 9262692: log likelihood = -3647.25
## AIC=7322.5    AICc=7323.63    BIC=7377.92
##
## Training set error measures:
##                  ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -16.71405 2991.913 2009.731 -3.68041 13.07944 0.3358039
##                  ACF1
## Training set -0.0006821151
```

```
tsdisplay(mod2_rid$residuals)
```

**mod2\_rid\$residuals**

```
checkresiduals(mod2_rid, test = FALSE)
```

#### Residuals from Regression with ARIMA(5,0,2)(1,0,1)[7] errors



```
checkresiduals(mod2_rid, test = "LB", lag = 25, plot = FALSE)
```

```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(5,0,2)(1,0,1)[7] errors  
## Q* = 27.451, df = 16, p-value = 0.03674  
##  
## Model df: 9. Total lags used: 25
```

```
pars_test(mod2_rid$coef, mod2_rid$var.coef)
```

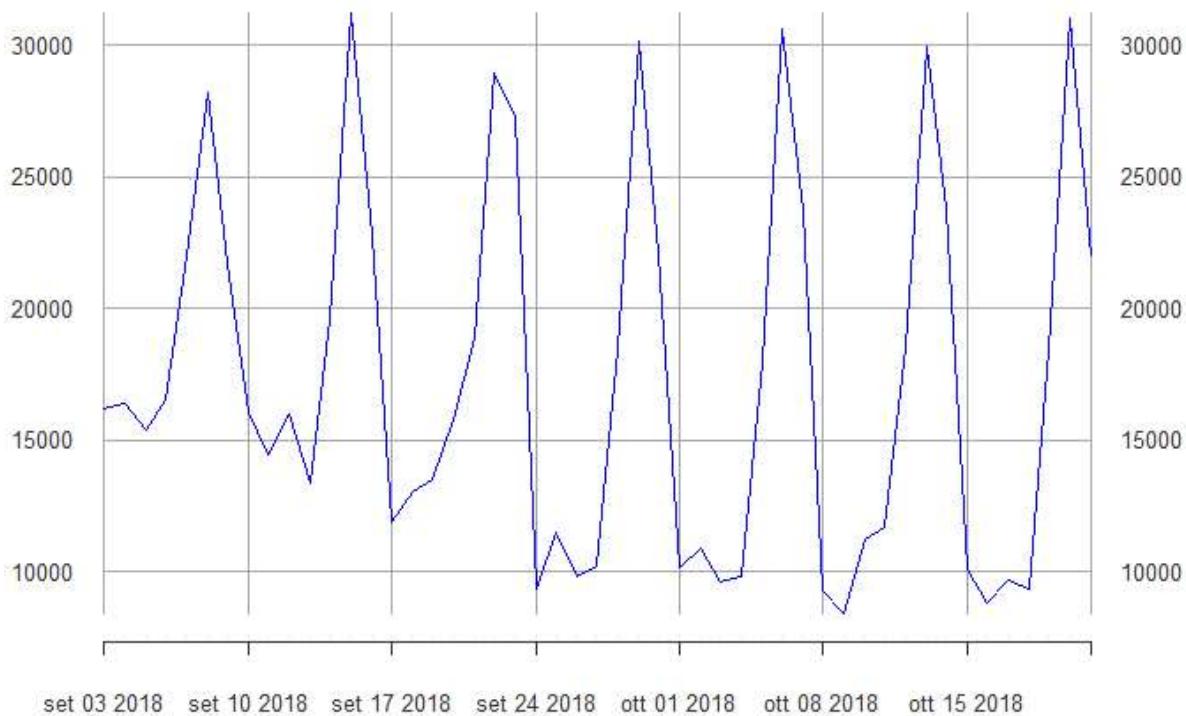
	ar1	ar2	ar3	ar4	ar5
##	3.958746e-09	9.241817e-03	5.466343e-01	4.909251e-01	3.953143e-01
##	ma1	ma2	sar1	sma1	intercept
##	9.149479e-06	5.897075e-02	0.000000e+00	0.000000e+00	0.000000e+00
##	Giorno_Friday	Giorno_Saturday	Giorno_Sunday		
##	9.037215e-13	0.000000e+00	0.000000e+00		

Le dummy stagionali ora risultano essere tutte significative, come ci aspettavamo. I parametri AR e MA non significativi verranno tolti nel prossimo modello, in quanto determinano anche un cambiamento nel comportamento dei residui che valuteremo successivamente. Procediamo quindi con il fit sul validation set.

```
plot(train[1:49,1],  
     col = "blue", lwd=0.5,  
     main = "Fitted Values")
```

**Fitted Values**

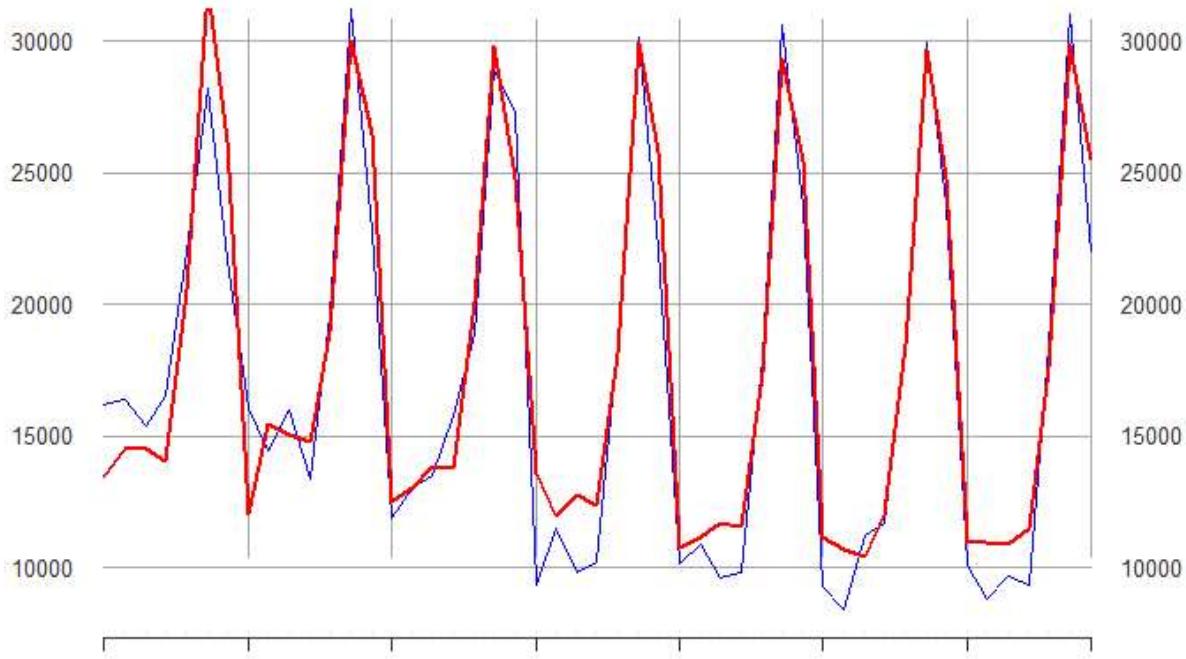
2018-09-03 / 2018-10-21



```
lines(xts(mod2_rid$fitted[1:49], order.by = index(train[1:49])), col="red", lwd=2.5)
```

**Fitted Values**

2018-09-03 / 2018-10-21



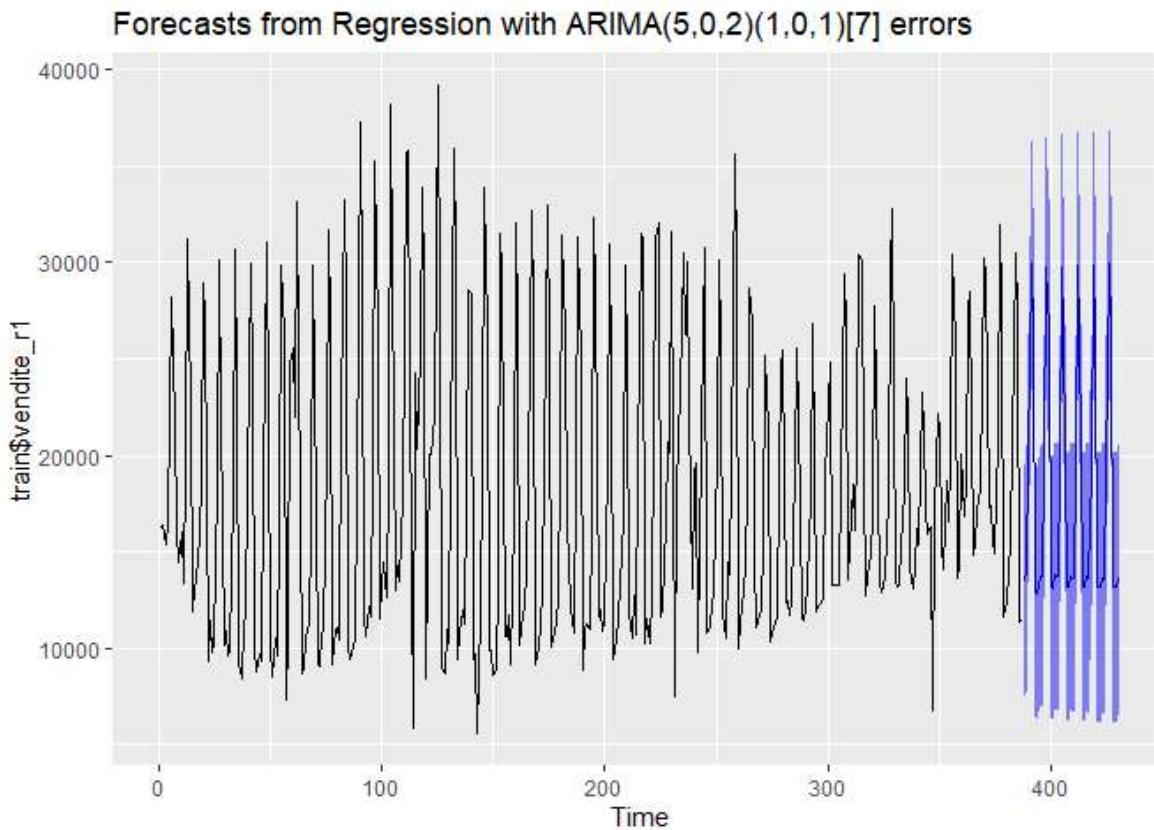
Anche in questo caso il modello riesce a spiegare bene i comportamenti stagionali a 7 giorni, ma per i comportamenti

non stagionali l'errore, anche se notiamo qualche piccolo miglioramento, rimane comunque notevole.

## Previsioni

```
# Al momento vengono fatte previsioni 44 passi in avanti (Lunghezza del validation set)
pred_mod2 <- forecast(mod2_rid, h = 44,
                      level = 95,
                      xreg = validation[, -c(1,5,6,7)])
```

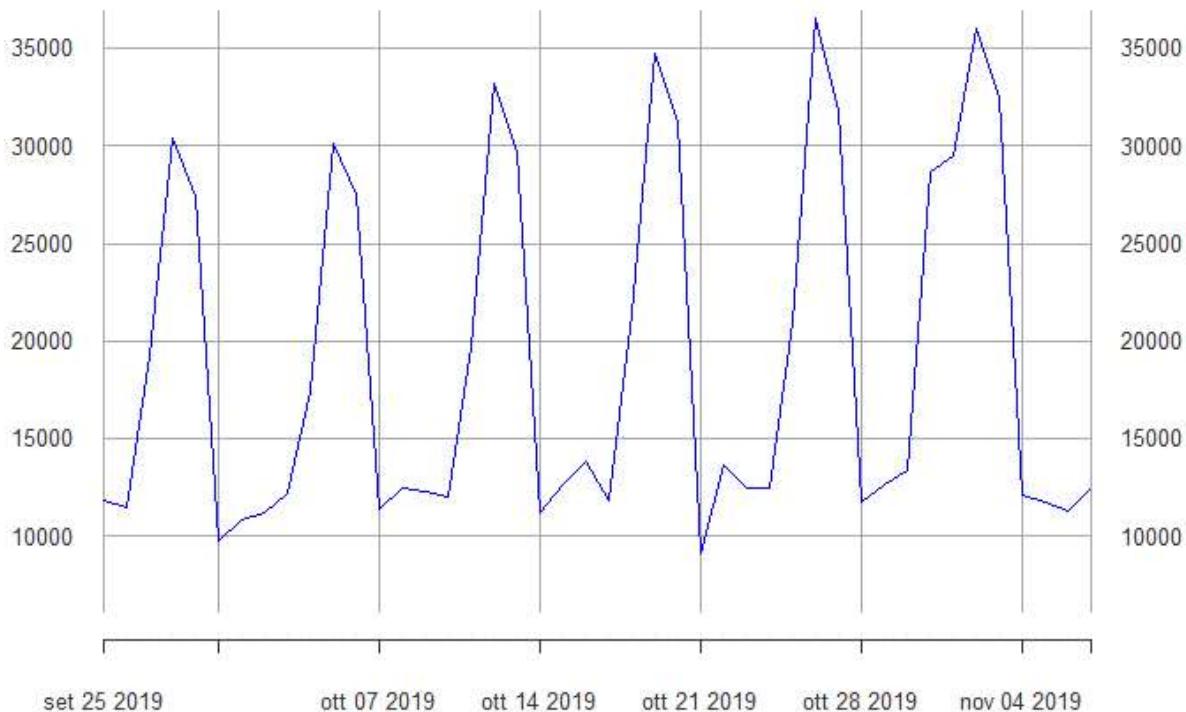
```
autoplot(pred_mod2)
```



```
plot(validation[,1],
      col = "blue", lwd=0.5,
      ylim = c(min(pred_mod2$lower), max(pred_mod2$upper)), main = "Fitted Value")
```

**Fitted Value**

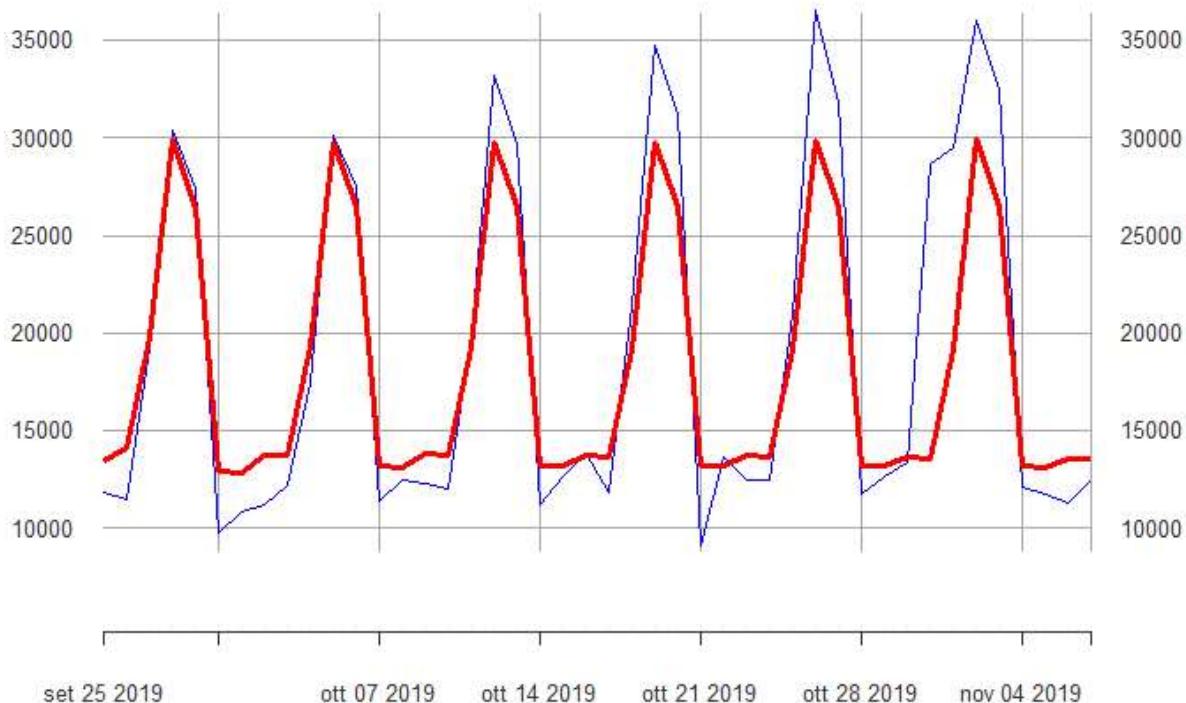
2019-09-25 / 2019-11-07



```
lines(xts(pred_mod2$mean, order.by = index(validation)), col="red", lwd=3)
```

**Fitted Value**

2019-09-25 / 2019-11-07



```
mape(validation[,1], xts(pred_mod2$mean, order.by = index(validation)))
```

```
## [1] 13.66721
```

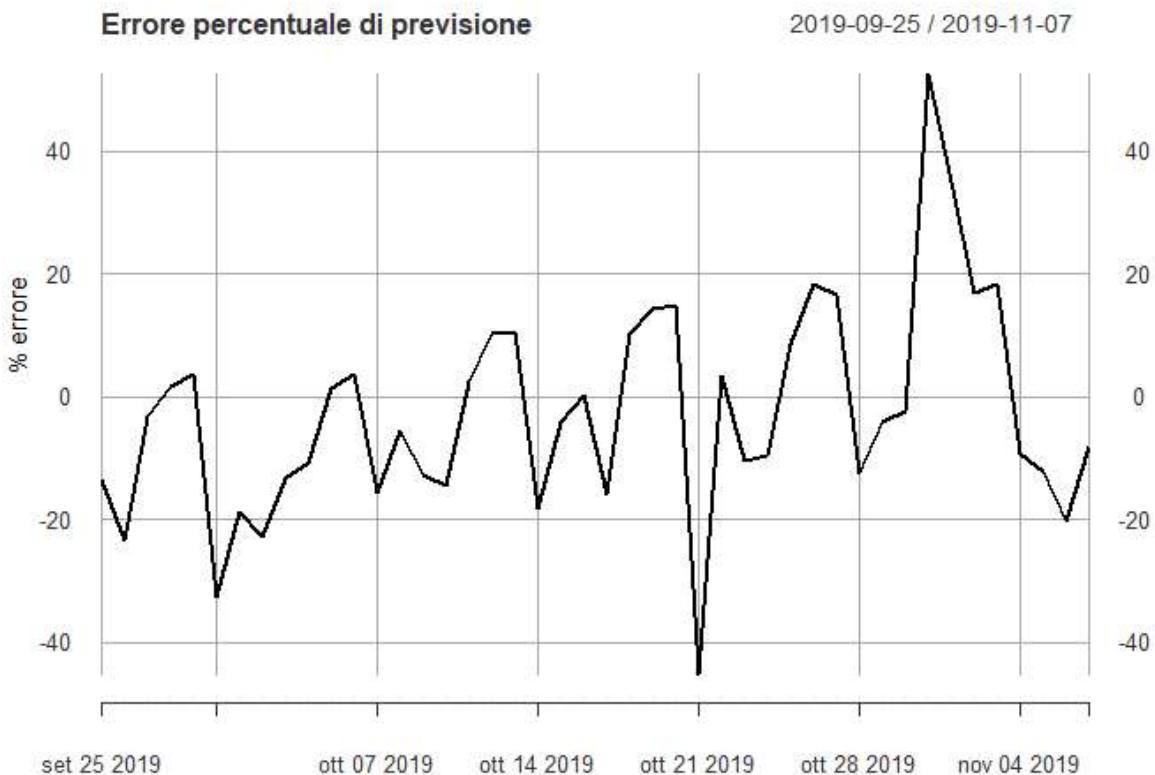
```
mae(validation[,1], xts(pred_mod2$mean, order.by = index(validation)))
```

```
## [1] 2604.205
```

```
rmse(validation[,1], xts(pred_mod2$mean, order.by = index(validation)))
```

```
## [1] 3827.902
```

```
err_plot(validation[,1], xts(pred_mod2$mean, order.by = index(validation)))
```



Sono state effettuate 44 previsioni con il modello considerato e sono state confrontate con il validation set.

Il MAE risulta pari a 2604.205, mentre l'RMSE risulta pari 2604.205. Se confrontiamo tali risultati con gli errori riscontrati sul training set, notiamo come è stato evitato il fenomeno di overfitting, anche se gli errori di previsioni (sia sul training che sul validation) rimangono comunque notevoli. Infine, nel complesso, non viene notato nessun miglioramento rispetto agli errori del modello1.

## Modello3

In questo modello andiamo ad escludere i parametri autoregressivi e a media mobile che nel modello 2 non sono risultati essere significativi. Nello specifico vengono considerati:

- Componente AR stagionale di ordine 1;
- Componente MA stagionale di ordine 1;
- **Componente MA non stagionale di ordine 3**
- **Componente AR non stagionale di ordine 4** (togliamo l'ultimo coefficiente autoregressivo per valutare se il terzo e il quarto diventano significativi. Nel caso non lo fossero verranno omessi in un eventuale *mod3\_rid*)
- Dummy come regressori esterni (solo quelle risultate significative)
- Costante inclusa

```
mod3 <- Arima(y = train$vendite_r1,
                 order = c(4, 0, 2),
                 list(order = c(1, 0, 1), period = 7),
                 xreg = train[, -c(1,5,6,7)],
                 include.constant = TRUE,
                 )
```

## Diagnostica

```
summary(mod3)
```

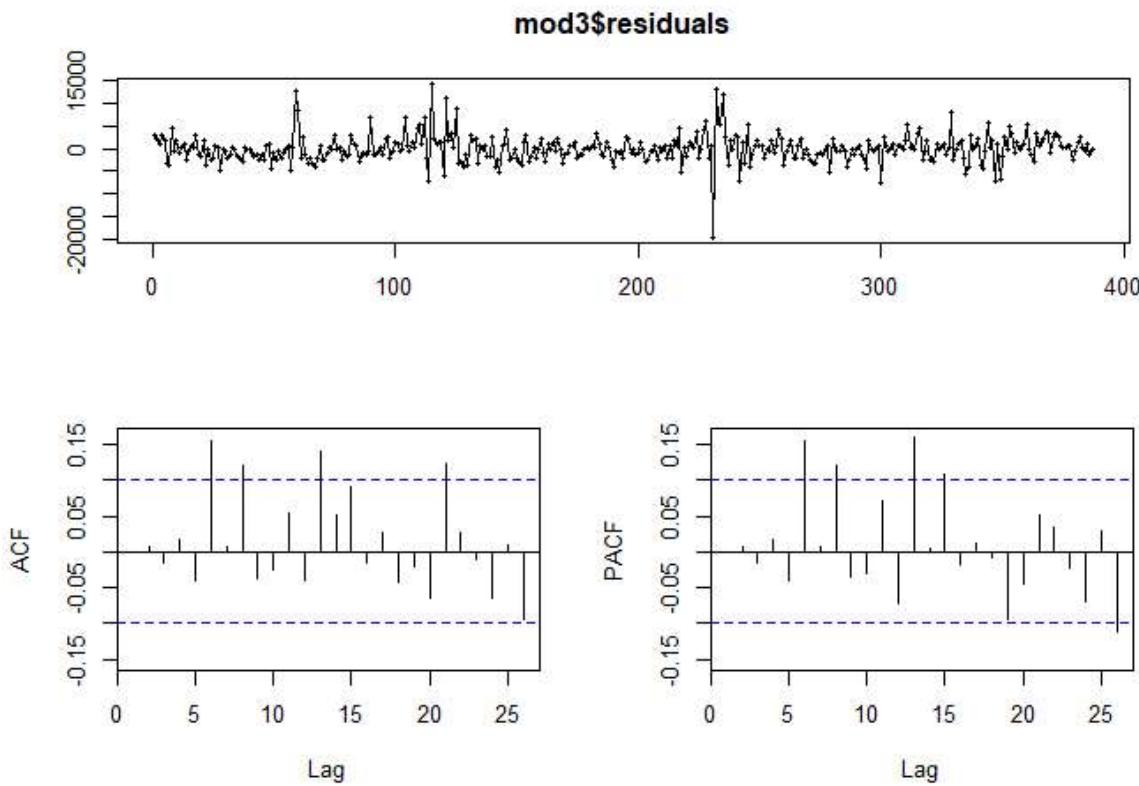
```
## Series: train$vendite_r1
## Regression with ARIMA(4,0,2)(1,0,1)[7] errors
##
## Coefficients:
##             ar1      ar2      ar3      ar4      ma1      ma2      sar1      sma1
##
```

```

##      -0.5702  -0.5694  0.3146  -0.0813  0.9673  0.9482  -0.1640  0.4311
## s.e.   0.0604   0.0771  0.0595   0.0573  0.0280  0.0530   0.2058  0.1908
##      intercept Giorno_Friday Giorno_Saturday Giorno_Sunday
##      12818.4177     6133.4524     17680.4320    13662.2802
## s.e.   348.2813     556.1246     605.0239    557.1159
##
## sigma^2 = 9620647: log likelihood = -3654.93
## AIC=7335.86   AICc=7336.84   BIC=7387.32
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -4.778231 3053.25 2070.51 -3.659588 13.37168 0.3459594
##                  ACF1
## Training set -6.320609e-05

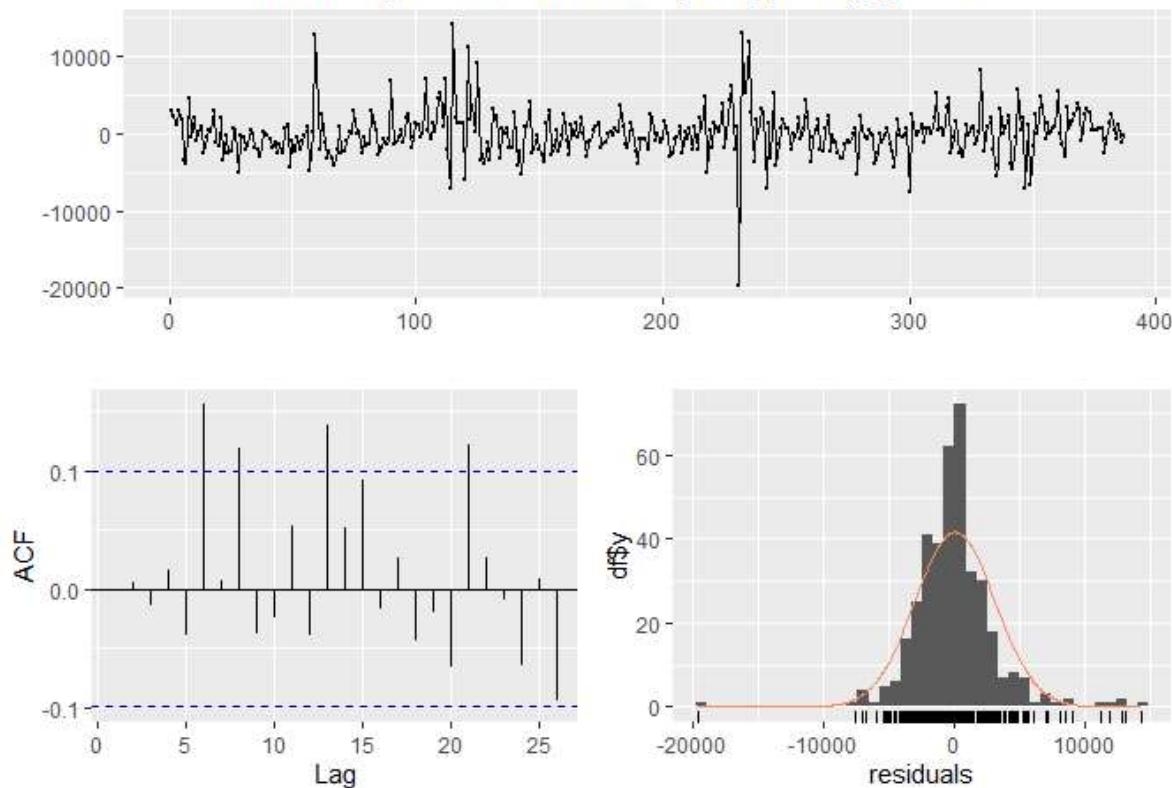
```

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



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

### Residuals from Regression with ARIMA(4,0,2)(1,0,1)[7] errors



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

```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(4,0,2)(1,0,1)[7] errors  
## Q* = 42.316, df = 17, p-value = 0.0006037  
##  
## Model df: 8. Total lags used: 25
```

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

```
##          ar1        ar2        ar3        ar4        ma1  
##  0.000000e+00 1.576517e-13 1.231411e-07 1.560261e-01 0.000000e+00  
##          ma2        sar1        sma1      intercept Giorno_Friday  
##  0.000000e+00 4.255124e-01 2.388812e-02 0.000000e+00 0.000000e+00  
## Giorno_Saturday Giorno_Sunday  
##  0.000000e+00 0.000000e+00
```

I parametri che **risultano essere significativi**, ad un livello  $\alpha = 0.05$ , sono:

- AR(1): *p-value*: 0.0000
- AR(2): *p-value*: 0.000
- AR(3): *p-value*: 0.000
- MA(1): *p-value*: 0.000
- MA(2): *p-value*: 0.000
- SMA(1)[7]: *p-value*: 0.023
- Giorno\_Friday *p-value*: 0.000
- Giorno\_Saturday *p-value*: 0.000
- Giorno\_Sunday *p-value*: 0.000

Mentre quelle che **non risultano essere significativi**:

- SAR(1)[7]: *p-value*: 0.425
- AR(4): *p-value*: 0.156

L'**AIC** risulta essere pari a : \$7335.86 \$

L'errore sul training set **MAE** risulta pari a 2070.51 mentre l'**RMSE** pari a: 3053.25

Il **Box-Ljung test** (*p-value* = 0.0006037) ci indica che questo modello presenta residui auto-correlati

Tale problema di autocorrelazione, potrebbe essere indotto dal fatto che non stiamo modellando la stagionalità a 365 giorni, la quale verrà inclusa nel successivo modello attraverso delle variabili dummy atte a modellare eventi annuali.

## Modello3 ridotto

Andiamo a togliere il parametro autoregressivo di ordine 4 mentre lasciamo il parametro autoregressivo stagionale. Nel caso non fosse ancora non significativo, si procederà ad ascludere anche

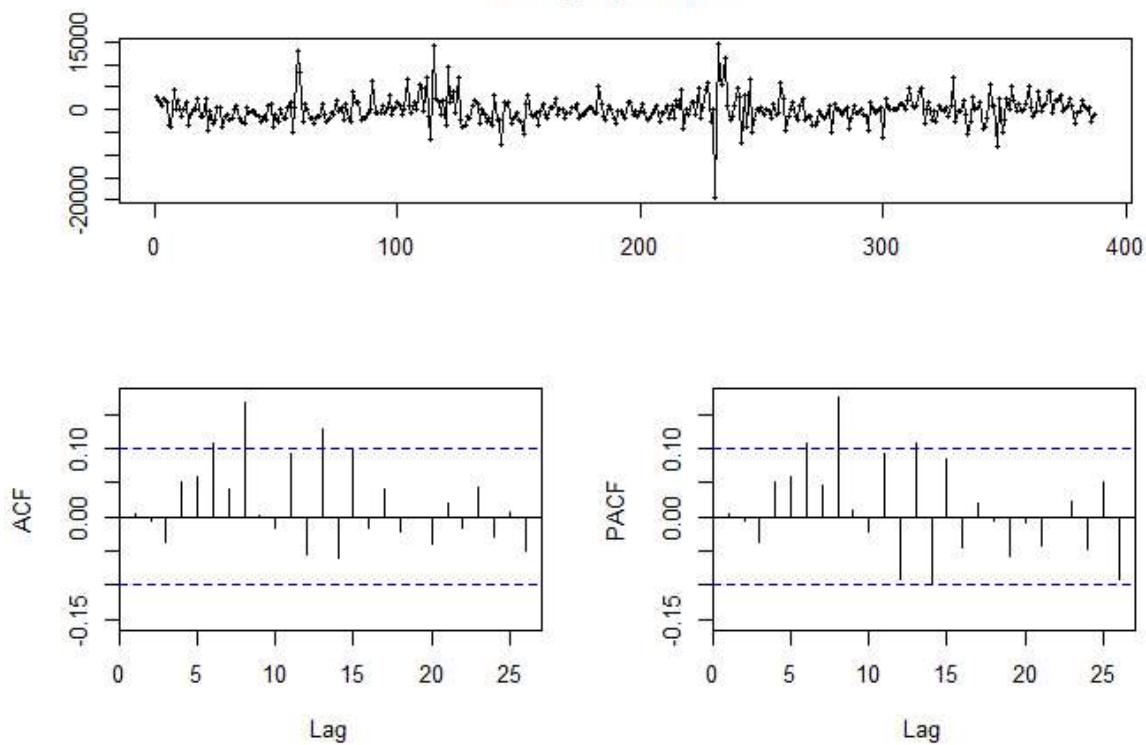
```
mod3_rid <- Arima(y = train$vendite_r1,
                     order = c(3, 0, 2),
                     list(order = c(1, 0, 1), period = 7),
                     xreg = train[, -c(1,5,6,7)],
```

```
  include.constant = TRUE,  
 )
```

```
summary(mod3_rid)
```

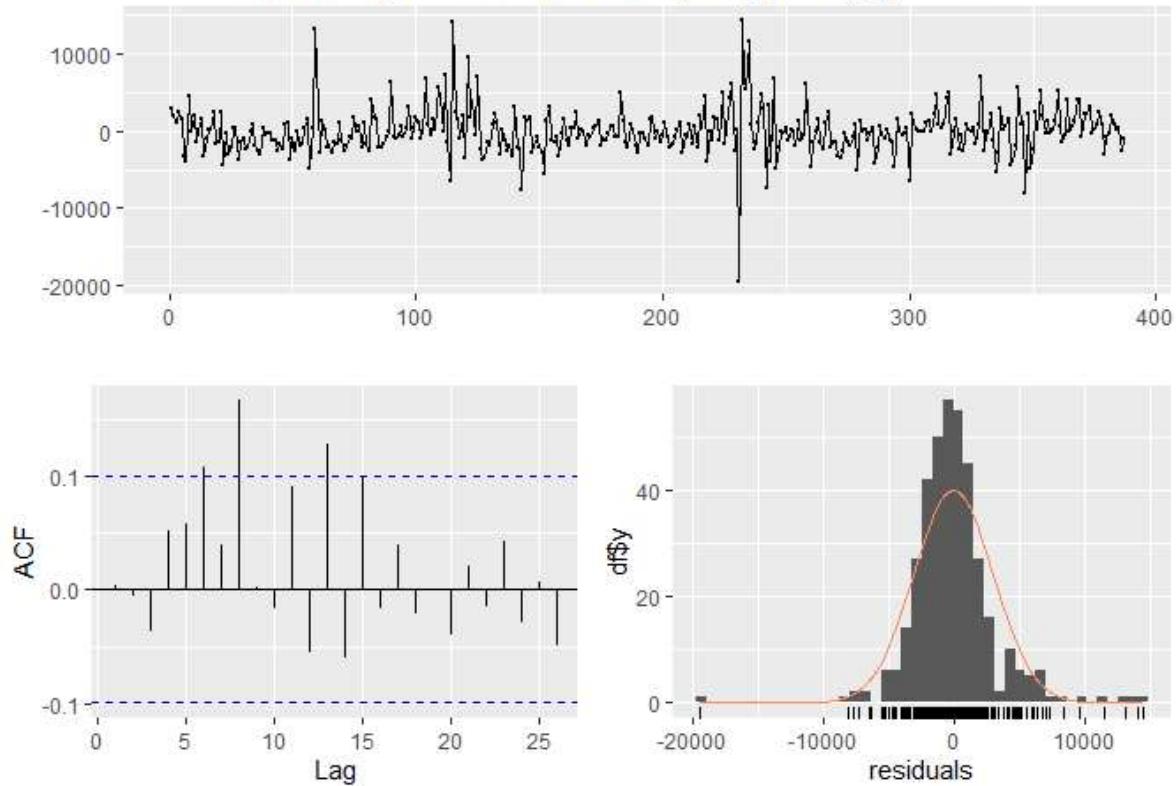
```
## Series: train$vendite_r1  
## Regression with ARIMA(3,0,2)(1,0,1)[7] errors  
##  
## Coefficients:  
##             ar1      ar2      ar3      ma1      ma2     sar1     sma1  intercept  
##            0.9853   -1.1370   0.3268  -0.5872   0.9459   0.8873  -0.7762 12940.2689  
## s.e.        0.0661    0.0737   0.0672   0.0385   0.0642   0.0620    0.0857   554.5028  
##             Giorno_Friday  Giorno_Saturday  Giorno_Sunday  
##                  6113.791       17463.9220      13436.384  
## s.e.          845.335        936.3355      851.068  
##  
## sigma^2 = 9467147: log likelihood = -3652.58  
## AIC=7329.16  AICc=7329.99  BIC=7376.66  
##  
## Training set error measures:  
##                 ME      RMSE      MAE      MPE      MAPE      MASE      ACF1  
## Training set -2.103891 3032.83 2041.045 -3.751676 13.3256 0.3410361 0.003418652
```

```
tsdisplay(mod3_rid$residuals)
```

**mod3\_rid\$residuals**

```
checkresiduals(mod3_rid, test = FALSE)
```

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



```
checkresiduals(mod3_rid, test = "LB", lag = 25, plot = FALSE)
```

```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(3,0,2)(1,0,1)[7] errors  
## Q* = 38.941, df = 18, p-value = 0.002902  
##  
## Model df: 7. Total lags used: 25
```

```
pars_test(mod3_rid$coef, mod3_rid$var.coef)
```

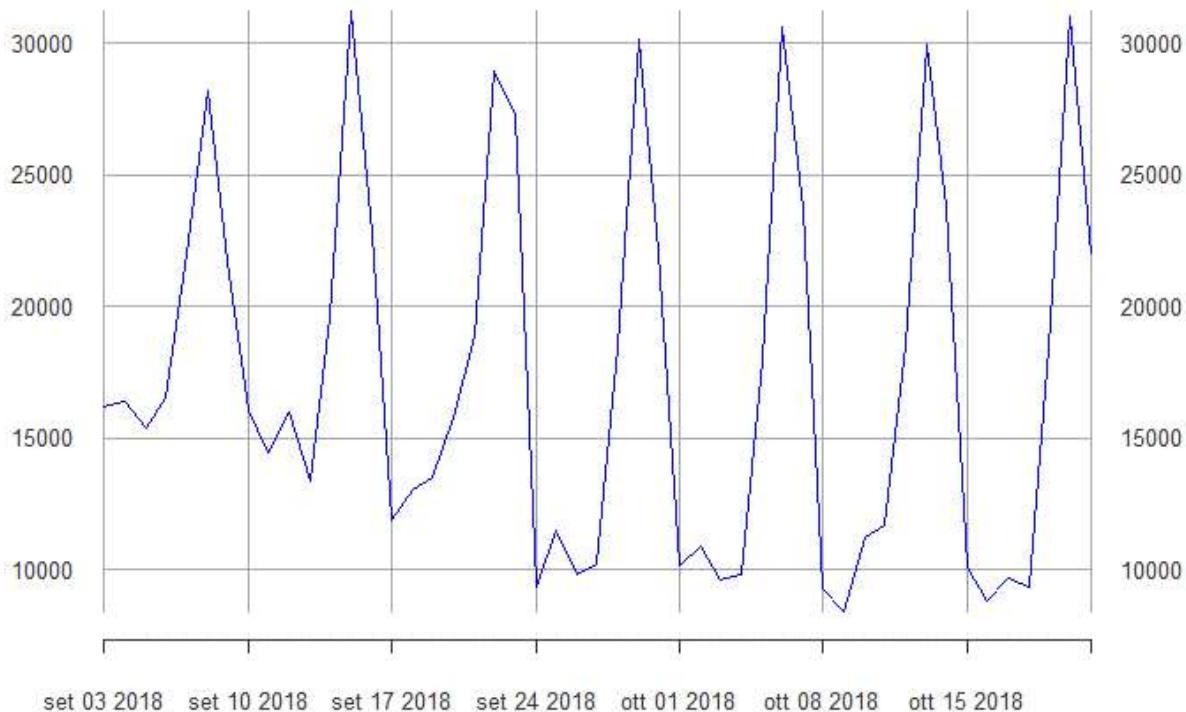
	ar1	ar2	ar3	ma1	ma2
##	0.000000e+00	0.000000e+00	1.162216e-06	0.000000e+00	0.000000e+00
##	sar1	sma1	intercept	Giorno_Friday	Giorno_Saturday
##	0.000000e+00	0.000000e+00	0.000000e+00	4.745093e-13	0.000000e+00
##	Giorno_Sunday				
##	0.000000e+00				

Il parametro autoregressivo stagionale risulta essere ora significativo. Procediamo quindi con il fit sul validation

```
plot(train[1:49,1],  
      col = "blue", lwd=0.5,  
      main = "Fitted Values")
```

**Fitted Values**

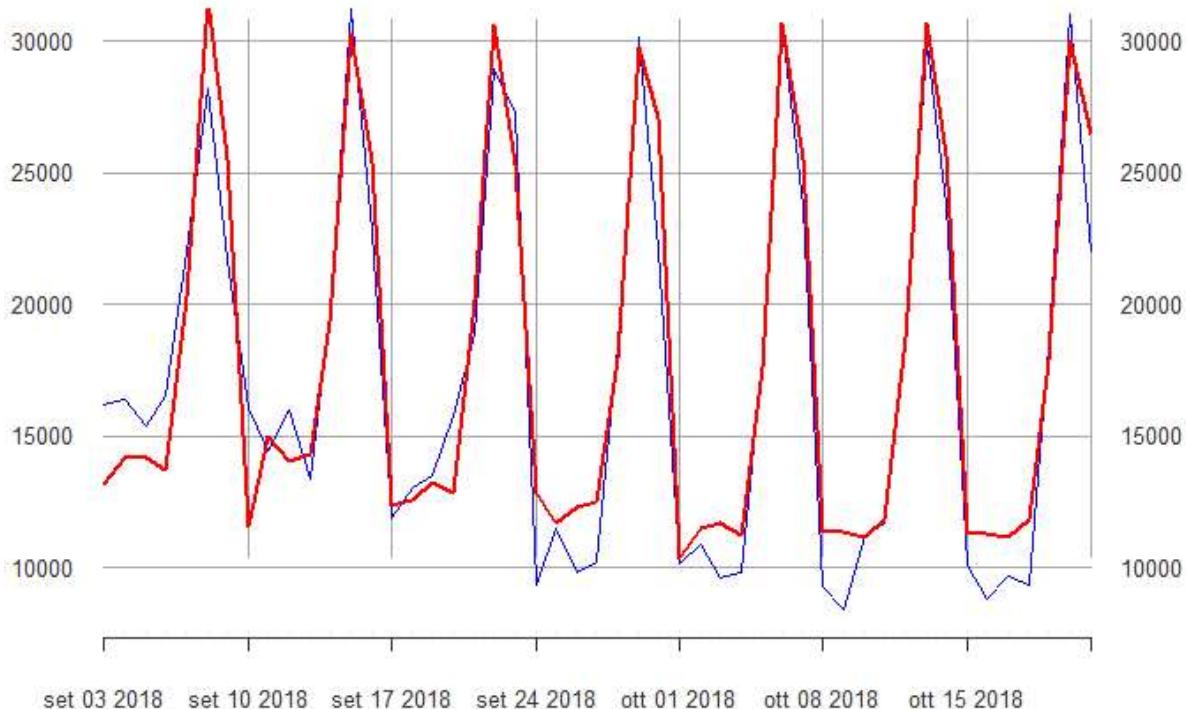
2018-09-03 / 2018-10-21



```
lines(xts(mod3$fitted[1:49], order.by = index(train[1:49])), col="red", lwd=2.5)
```

**Fitted Values**

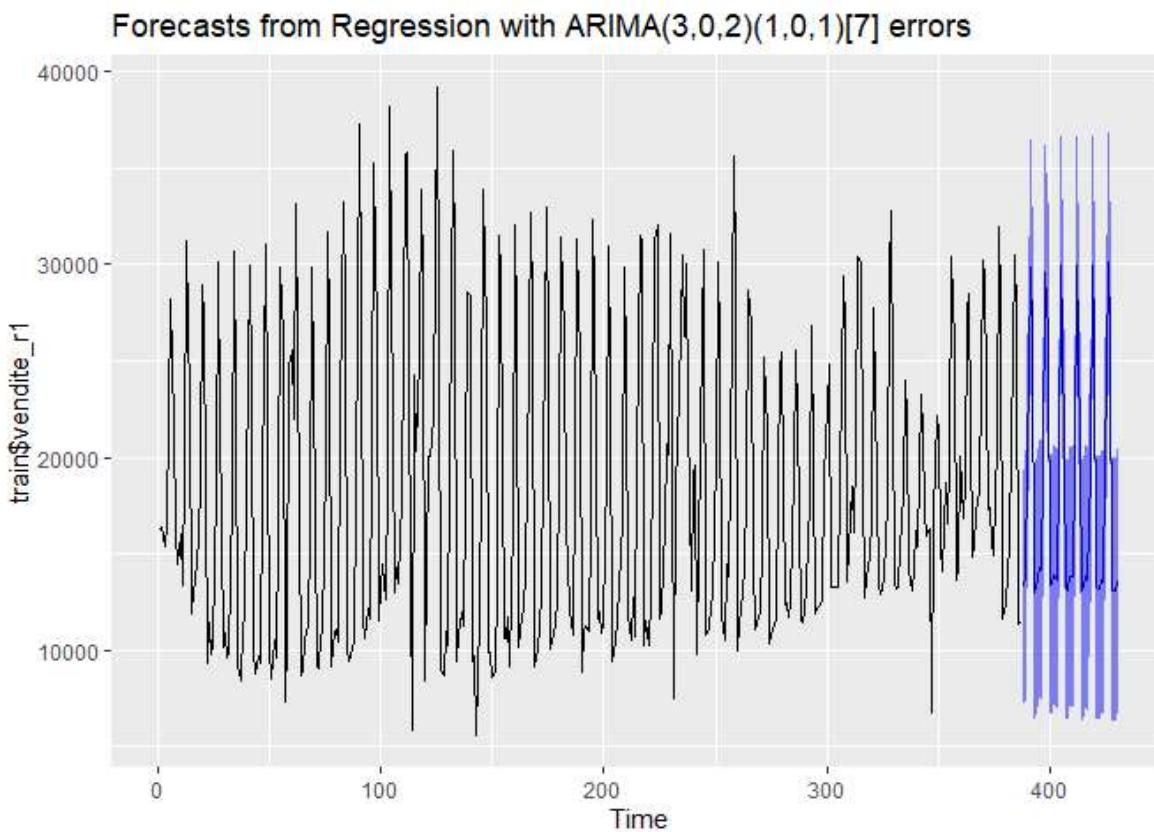
2018-09-03 / 2018-10-21



## Previsioni

```
# Al momento vengono fatte previsioni 44 passi in avanti (Lunghezza del validation set)
pred_mod3 <- forecast(mod3_rid, h = 44,
                      level = 95,
                      xreg = validation[, -c(1,5,6,7)])
```

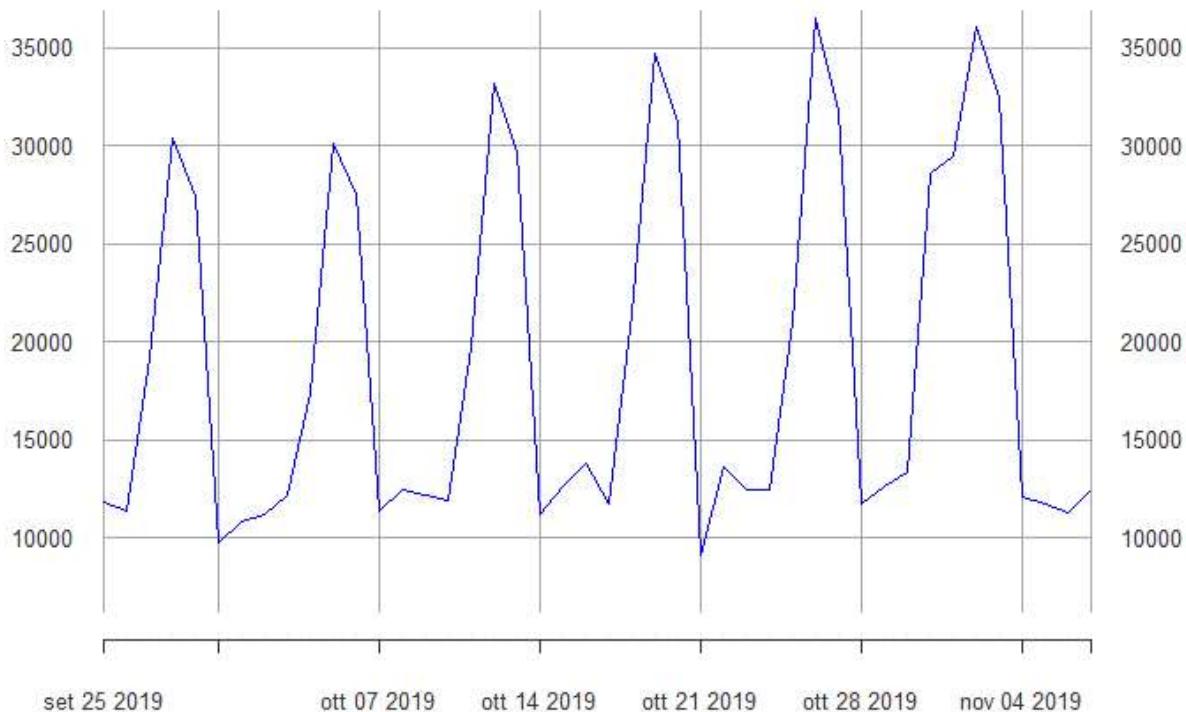
```
autoplot(pred_mod3)
```



```
plot(validation[,1],
      col = "blue", lwd=0.5,
      ylim = c(min(pred_mod3$lower), max(pred_mod3$upper)), main = "Fitted Value")
```

**Fitted Value**

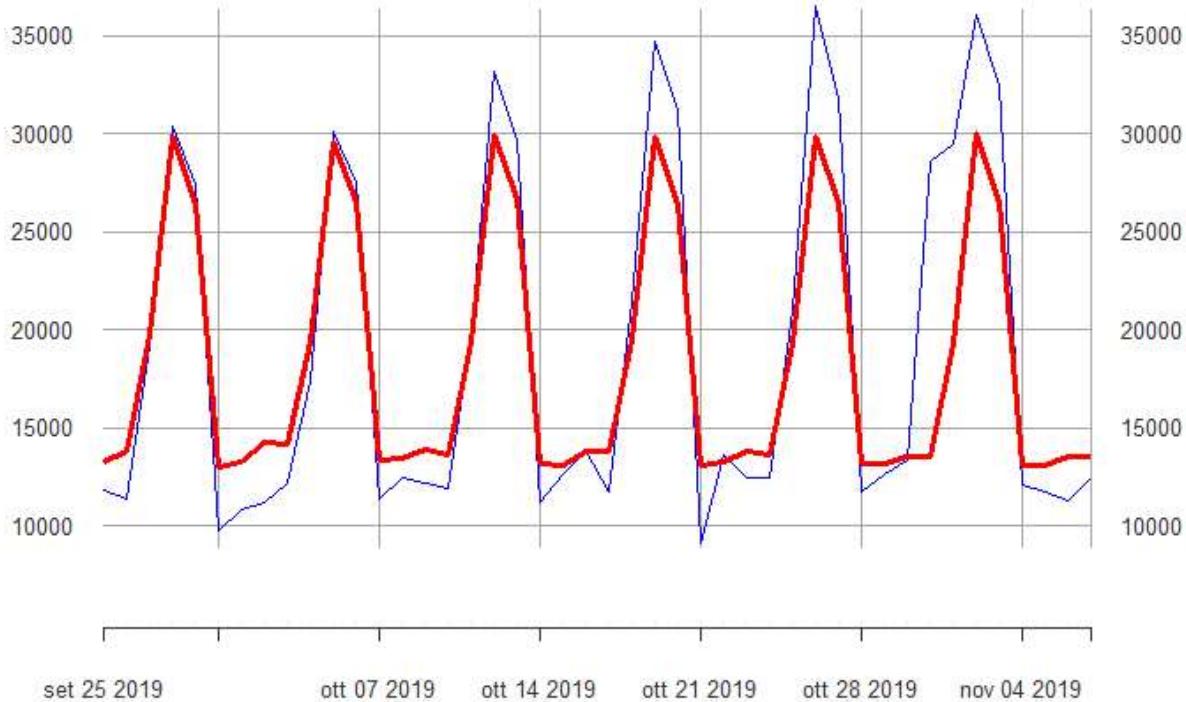
2019-09-25 / 2019-11-07



```
lines(xts(pred_mod3$mean, order.by = index(validation)), col="red", lwd=3)
```

**Fitted Value**

2019-09-25 / 2019-11-07



```
mape(validation[,1], xts(pred_mod3$mean, order.by = index(validation)))
```

```
## [1] 13.90544
```

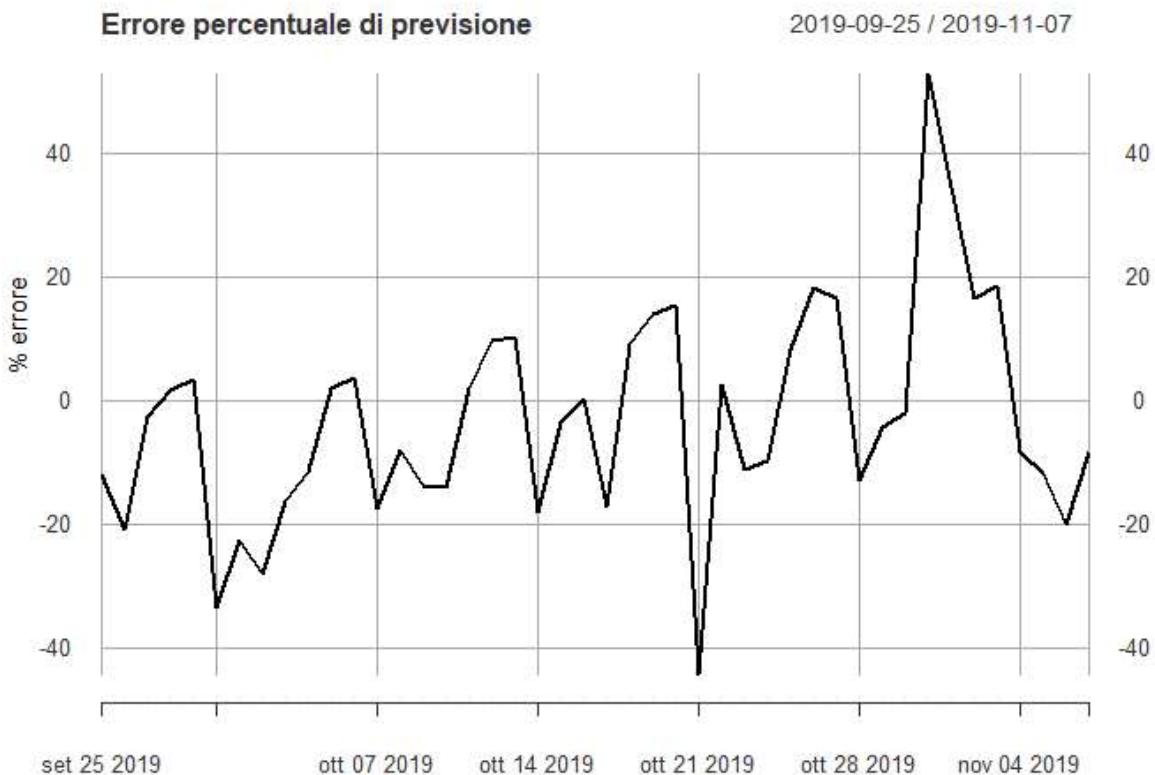
```
mae(validation[,1], xts(pred_mod3$mean, order.by = index(validation)))
```

```
## [1] 2618.055
```

```
rmse(validation[,1], xts(pred_mod3$mean, order.by = index(validation)))
```

```
## [1] 3822.795
```

```
err_plot(validation[,1], xts(pred_mod3$mean, order.by = index(validation)))
```



Sono state effettuate 44 previsioni con il modello considerato e sono state confrontate con il validation set.

Il MAE risulta pari a **2618.055**, mentre l'RMSE risulta pari **3822.795**. Se confrontiamo tali risultati con gli errori riscontrati sul training set, notiamo come è stato evitato il fenomeno di overfitting, anche se gli errori di previsioni (sia sul training che sul validation) rimangono comunque notevoli. Infine, nel complesso, viene notato un peggioramento rispetto agli errori del modello2.

## Considerazioni sui primi tre modelli

I primi tre modelli considerati presentano svariati problemi:

- Tutti i modelli presentano residui autocorrelati, come suggerito dal test di Ljung-Box il quale ci indica che molta informazione è ancora inclusa all'interno dei residui.
- Tutti i modelli presentano alti valori sia di MAE che di RMSE.
- Osservando i fit, sia sul training che sul test, sembra che la stagionalità a 7 giorni venga ben interpretata.

Tali problemi sono dovuti principalmente al fatto che non stiamo modellando la stagionalità annuale del modello (i.e.: Festività quali Natale, Ferragosto ecc..) con conseguenti stime pessime per tali andamenti.

## Modello 4 SARIMAX

Consideriamo quindi un modello SARIMAX, dove:

- Componenti ARIMA: consideriamo quella del **modello3** (dove i parametri risultavano tutti significativi) SARIMA(4,0,2)(1,0,1)[7]
- Stagionalità settimanale: dummy **giorni settimanali** (Martedì-Domenica, Lunedì sempre escluso)
- Stagionalità annuale: variabili dummy per modellare tutti quegli andamenti che si riferiscono a stagionalità annuali: Natale, Halloween, Ferragosto ecc...
- Altri regressori: i.e. costo del riscaldamento in litri, precipitazioni giornaliere ecc..
- Costante INCLUSA

## Pre-processing

```
#Stagionalità annuale
fest <- read_xlsx("../\\Dati aggiuntivi\\fest.xlsx", col_types = NULL)
```

```
str(fest)
```

```
## # tibble [539 x 17] (S3: tbl_df/tbl/data.frame)
## # $ date      : POSIXct[1:539], format: "2018-09-03" "2018-09-04" ...
## # $ dec24     : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ dec25     : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ dec26     : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ jan1      : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ jan6      : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ aug15     : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ dec31     : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ eastsun   : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ eastermon : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ oct31     : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ nov1      : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ apr25     : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ mag1      : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ jun2      : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ martgrasso: num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
## # $ bridge    : num [1:539] 0 0 0 0 0 0 0 0 0 0 ...
```

# Unisco al df contenenti le dummy settimanali con quello contenenti le dummy annuali

```
xreg <- cbind(dum_day_rid, fest)
xreg <- subset(xreg, select = -c(date))
head(xreg)
```

	<b>data</b>	<b>Giorno_Friday</b>	<b>Giorno_Saturday</b>	<b>Giorno_Sunday</b>	<b>Giorno_Thursday</b>
	<date>	<int>	<int>	<int>	<int>
1	2018-09-03	0	0	0	0
2	2018-09-04	0	0	0	0
3	2018-09-05	0	0	0	0
4	2018-09-06	0	0	0	1
5	2018-09-07	1	0	0	0
6	2018-09-08	0	1	0	0

6 rows | 1-6 of 24 columns

```
r1$Pioggia[r1$Pioggia==" "] <- "False"
```

```
require(fastDummies)
dum_pioggia <- r1[, c("data", "Pioggia")]
dum_pioggia[, "Pioggia"] <- as.factor(dum_pioggia$Pioggia)
dum_pioggia <- dummy_cols(dum_pioggia, select_columns = c("Pioggia"),
                           remove_first_dummy = TRUE,
                           remove_selected_columns = TRUE)
```

```
xreg$Riscaldamento <- r1$GASOLIO_RISCALDAMENTO.LITRO
xreg$Pioggia_True <- dum_pioggia$Pioggia_True
```

```
xreg <- xts(xreg[, -1], xreg$data)
```

```
colnames(xreg)
```

```
## [1] "Giorno_Friday"      "Giorno_Saturday"     "Giorno_Sunday"      "Giorno_Thursday"
## [5] "Giorno_Tuesday"     "Giorno_Wednesday"    "dec24"              "dec25"
## [9] "dec26"                "jan1"                "jan6"               "aug15"
## [13] "dec31"                "eastsun"             "eastermon"         "oct31"
## [17] "nov1"                 "apr25"               "mag1"               "jun2"
## [21] "martgrasso"          "bridge"              "Riscaldamento"     "Pioggia_True"
```

```
dim(xreg)
```

```
## [1] 539 24
```

```
df <- cbind(vendite_r1, xreg)
head(df)
```

	vendite_r1	Giorno_Friday	Giorno_Saturday	Giorno_Sunday
## 2018-09-03	16201.27	0	0	0
## 2018-09-04	16409.71	0	0	0
## 2018-09-05	15374.10	0	0	0
## 2018-09-06	16531.07	0	0	0

```

## 2018-09-07 22057.70      1      0      0
## 2018-09-08 28190.46      0      1      0
## Giorno_Thursday Giorno_Tuesday Giorno_Wednesday dec24 dec25 dec26
## 2018-09-03      0      0      0      0      0      0
## 2018-09-04      0      1      0      0      0      0
## 2018-09-05      0      0      1      0      0      0
## 2018-09-06      1      0      0      0      0      0
## 2018-09-07      0      0      0      0      0      0
## 2018-09-08      0      0      0      0      0      0
## jan1 jan6 aug15 dec31 eastsun eastermon oct31 nov1 apr25 mag1 jun2
## 2018-09-03 0 0 0 0 0 0 0 0 0 0 0 0
## 2018-09-04 0 0 0 0 0 0 0 0 0 0 0 0
## 2018-09-05 0 0 0 0 0 0 0 0 0 0 0 0
## 2018-09-06 0 0 0 0 0 0 0 0 0 0 0 0
## 2018-09-07 0 0 0 0 0 0 0 0 0 0 0 0
## 2018-09-08 0 0 0 0 0 0 0 0 0 0 0 0
## martgrasso bridge Riscaldamento Pioggia_True
## 2018-09-03 0 0 1.30325 0
## 2018-09-04 0 0 1.30325 0
## 2018-09-05 0 0 1.30325 0
## 2018-09-06 0 0 1.30325 1
## 2018-09-07 0 0 1.30325 0
## 2018-09-08 0 0 1.30325 0

```

```
write.zoo(df, file="..\Dati_aggiuntivi\Dat CV_Arima\df_4.csv", sep=",")
```

```
colnames(df)
```

```

## [1] "vendite_r1"      "Giorno_Friday"    "Giorno_Saturday"  "Giorno_Sunday"
## [5] "Giorno_Thursday" "Giorno_Tuesday"   "Giorno_Wednesday" "dec24"
## [9] "dec25"           "dec26"          "jan1"            "jan6"
## [13] "aug15"           "dec31"          "eastsun"         "eastermon"
## [17] "oct31"           "nov1"           "apr25"          "mag1"
## [21] "jun2"            "martgrasso"     "bridge"          "Riscaldamento"
## [25] "Pioggia_True"
```

```
dim(df)
```

```
## [1] 539 25
```

## Partizionamento del dataset

Rifaccio la divisione trainin/validation/test set con il dataset xreg appena creato.

```
# Divisione training-test set
train_date <- nrow(df) *0.8
train_temp <- df[1:train_date,]
test <- df[-c(1:train_date),] # Usare alla fine
```

```
# Training - validation set
train_date_rid <- nrow(train_temp)*0.9
train<- train_temp[1:train_date_rid,]
validation<- train_temp[-c(1:train_date_rid),]
```

## Stima del modello

```
mod4 <- Arima(y = train$vendite_r1,
                order = c(3, 0, 2),
                list(order = c(1, 0, 1), period = 7),
                xreg = train[, -1],
                include.constant = TRUE,
                )
```

```
summary(mod4)
```

```
## Series: train$vendite_r1
## Regression with ARIMA(3,0,2)(1,0,1)[7] errors
##
## Coefficients:
##             ar1      ar2      ar3      ma1      ma2      sar1      sma1  intercept
##             -0.2139  -0.6371  0.4159  0.6458  1.0000  0.8130  -0.5285  55232.24
## s.e.      0.0654   0.0432  0.0629  0.0145  0.0105  0.1013   0.1722  15735.97
##             Giorno_Friday  Giorno_Saturday  Giorno_Sunday  Giorno_Thursday
##                 7295.716       18394.1465      14284.2847      1408.688
## s.e.      1061.103        952.5006      823.7851      1061.222
```

```

##      Giorno_Tuesday Giorno_Wednesday      dec24      dec25      dec26
##      515.0762       1234.6928 -2019.200 -10391.901  5618.631
## s.e.    808.5650        944.8714  2211.994   2254.753 2154.744
##      jan1       jan6     aug15     dec31    eastsun eastermon
##      5069.129 -2113.256 -7857.959 -3258.032 -21724.722 2165.745
## s.e.  2109.858  1941.842  1992.711  2022.939  2007.332 2193.855
##      oct31      nov1     apr25     mag1     jun2 martgrasso
##      12435.647 13701.000 13234.534  5626.454 -1595.881  420.6642
## s.e. 2030.661  2082.941  2180.424  1964.396  1921.866 1936.0212
##      bridge Riscaldamento Pioggia_True
##      1804.004 -33410.57   1300.8306
## s.e. 1002.131   12060.58   281.0915
##
## sigma^2 = 5284364: log likelihood = -3531.27
## AIC=7128.55  AICc=7134.9  BIC=7259.17
##
## Training set error measures:
##          ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -26.91243 2201.684 1677.718 -2.148785 10.59169 0.2803281
##          ACF1
## Training set 0.00910854

```

**pars\_test(mod4\$coef, mod4\$var.coef)**

```

##          ar1          ar2          ar3          ma1
## 1.065591e-03 0.000000e+00 3.672573e-11 0.000000e+00
##          ma2          sar1          sma1      intercept
## 0.000000e+00 8.881784e-16 2.147409e-03 4.482159e-04
## Giorno_Friday Giorno_Saturday Giorno_Sunday Giorno_Thursday
## 6.173062e-12 0.000000e+00 0.000000e+00 1.843696e-01
## Giorno_Tuesday Giorno_Wednesday      dec24      dec25
## 5.241085e-01 1.913041e-01 3.613260e-01 4.048302e-06
##          dec26          jan1          jan6      aug15
## 9.118925e-03 1.627933e-02 2.764740e-01 8.035078e-05
##          dec31          eastsun      eastermon      oct31
## 1.072791e-01 0.000000e+00 3.235509e-01 9.128991e-10
##          nov1          apr25          mag1      jun2
## 4.777201e-11 1.281450e-09 4.180429e-03 4.063235e-01
##      martgrasso      bridge Riscaldamento Pioggia_True
## 8.279879e-01 7.183433e-02 5.601675e-03 3.695991e-06

```

Come già riscontrato in precedenza le variabili che modellano i giorni della settimana, risultano essere non significative:

- Giorno\_Thursday *p-value*: 0.265
- Giorno\_Tuesday *p-value*: 0.693
- Giorno\_Wednesday *p-value*: 0.279

Cominciamo quindi togliendo quelle.

```
mod4_rid1 <- Arima(y = train$vendite_r1,
                     order = c(3, 0, 2),
                     list(order = c(1, 0, 1), period = 7),
                     xreg = train[, -c(1,5,6,7)],
                     include.constant = TRUE,
                     )
```

```
summary(mod4_rid1)
```

```
## Series: train$vendite_r1
## Regression with ARIMA(3,0,2)(1,0,1)[7] errors
##
## Coefficients:
##             ar1      ar2      ar3      ma1      ma2      sar1      sma1  intercept
##             -0.2075  -0.6329  0.4244  0.6464  1.0000  0.8421  -0.5668  57083.75
## s.e.      0.0634   0.0424  0.0597  0.0145  0.0105  0.0801   0.1441  15612.75
##             Giorno_Friday  Giorno_Saturday  Giorno_Sunday      dec24      dec25
##             6220.7248     17661.01    13855.9405  -2096.387  -10416.971
## s.e.      797.7469      851.50    809.1284   2208.410   2255.422
##             dec26      jan1      jan6      aug15      dec31      eastsun  eastermon
##             5642.364  4997.511  -2108.322  -7767.042  -3313.479  -21824.41  2052.863
## s.e.     2157.480  2099.405   1939.091   1988.602   2019.290   2004.28  2188.758
##             oct31      nov1      apr25      mag1      jun2  martgrasso
##             12512.549 13814.932  13303.564  5752.359  -1547.510    490.7224
## s.e.     2027.776  2082.411   2176.285  1946.320   1917.056   1923.5675
##             bridge  Riscaldamento  Pioggia_True
##             1818.729     -34230.11    1309.4381
## s.e.     1001.285      11976.84    279.8033
##
## sigma^2 = 5263159: log likelihood = -3532.27
## AIC=7124.54  AICc=7129.77  BIC=7243.3
##
```

```
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set -26.27932 2206.527 1686.764 -2.04933 10.65686 0.2818396 0.00930651
```

```
pars_test(mod4_rid1$coef, mod4_rid1$var.coef)
```

	ar1	ar2	ar3	ma1	ma2
##	1.068214e-03	0.000000e+00	1.204814e-12	0.000000e+00	0.000000e+00
##	sar1	sma1	intercept	Giorno_Friday	Giorno_Saturday
##	0.000000e+00	8.418011e-05	2.559562e-04	6.217249e-15	0.000000e+00
##	Giorno_Sunday	dec24	dec25	dec26	jan1
##	0.000000e+00	3.424811e-01	3.862729e-06	8.916048e-03	1.729191e-02
##	jan6	aug15	dec31	eastsun	eastermon
##	2.769160e-01	9.392215e-05	1.008155e-01	0.000000e+00	3.482894e-01
##	oct31	nov1	apr25	mag1	jun2
##	6.804060e-10	3.264788e-11	9.779415e-10	3.121574e-03	4.195323e-01
##	martgrasso	bridge	Riscaldamento	Pioggia_True	
##	7.986377e-01	6.930980e-02	4.262874e-03	2.870817e-06	

Le variabili che risultano essere maggiormente non significative sono:

- dec24 *p-value*: 0.29 8
- jun2 *p-value*: 0.31 21
- martgrasso *p-value*: 0.81

Ristiamo il modello togliendo le variabili maggiormente non significative

```
mod4_rid2<- Arima(y = train$vendite_r1,
                  order = c(3, 0, 2),
                  list(order = c(1, 0, 1), period = 7),
                  xreg = train[, -c(1,5,6,7,8,21,22)],
                  include.constant = TRUE,
                  )
```

```
summary(mod4_rid2)
```

```
## Series: train$vendite_r1
## Regression with ARIMA(3,0,2)(1,0,1)[7] errors
```

```

## 
## Coefficients:
##             ar1      ar2      ar3      ma1      ma2      sar1      sma1  intercept
##             -0.2103  -0.6358  0.4081  0.6424  1.0000  0.8412  -0.5631  57337.34
## s.e.      0.0637   0.0419  0.0571  0.0144  0.0104  0.0771   0.1386 15350.30
##             Giorno_Friday  Giorno_Saturday  Giorno_Sunday      dec25      dec26
##             6220.239     17662.7857    13844.0838  -9726.432  6078.299
## s.e.      806.644      858.6649    816.2785   2050.376 2033.844
##             jan1      jan6     aug15     dec31     eastsun  eastermon
##             4913.412  -2115.882  -7565.922  -3120.521 -21607.341 2006.414
## s.e.  2089.628   1914.823   2005.047   1998.329   1975.252 2200.727
##             oct31      nov1     apr25     mag1      bridge  Riscaldamento
##             12675.373 13892.164 13296.903  5810.295  1920.540  -34431.63
## s.e.  2015.227  2072.066  2165.399  1919.733   984.186 11776.17
##             Pioggia_True
##             1278.9635
## s.e.  277.5043
## 
## sigma^2 = 5239034: log likelihood = -3533.05
## AIC=7120.11  AICc=7124.32  BIC=7226.99
## 
## Training set error measures:
##             ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -25.45422 2210.669 1695.667 -2.041842 10.69313 0.2833272
##             ACF1
## Training set 0.00834851

```

**pars\_test(mod4\_rid2\$coef, mod4\_rid2\$var.coef)**

	ar1	ar2	ar3	ma1	ma2
##	9.556183e-04	0.000000e+00	9.128254e-13	0.000000e+00	0.000000e+00
##	sar1	sma1	intercept	Giorno_Friday	Giorno_Saturday
##	0.000000e+00	4.843076e-05	1.875229e-04	1.243450e-14	0.000000e+00
##	Giorno_Sunday	dec25	dec26	jan1	jan6
##	0.000000e+00	2.098181e-06	2.802806e-03	1.870628e-02	2.691589e-01
##	aug15	dec31	eastsun	eastermon	oct31
##	1.610130e-04	1.183905e-01	0.000000e+00	3.619241e-01	3.178786e-10
##	nov1	apr25	mag1	bridge	Riscaldamento
##	2.021006e-11	8.219758e-10	2.473082e-03	5.100959e-02	3.457420e-03
##	Pioggia_True				
##	4.049891e-06				

Le uniche due variabile , con un  $\alpha = 0.05$ , non risultano significative sono *eastermon* e *jan 6* con un *p-value* pari a 0.1055 e 0.269. Escludiamo quindi tali variabile e ristiamo il modello.

```
mod4_rid3<- Arima(y = train$vendite_r1,
                     order = c(3, 0, 2),
                     list(order = c(1, 0, 1), period = 7),
                     xreg = train[, -c(1,5,6,7,8,12,16,21,22)],
                     include.constant = TRUE,
                     )
```

```
summary(mod4_rid3)
```

```
## Series: train$vendite_r1
## Regression with ARIMA(3,0,2)(1,0,1)[7] errors
##
## Coefficients:
##             ar1      ar2      ar3      ma1      ma2     sar1     sma1 intercept
##             -0.1936 -0.6281  0.4033  0.6379  1.0000  0.8501 -0.5882  56673.95
## s.e.      0.0613  0.0412  0.0568  0.0137  0.0104  0.0735  0.1317  15121.50
##             Giorno_Friday Giorno_Saturday Giorno_Sunday      dec25      dec26
##             6192.4718    17635.8838   13800.1401 -9688.808  6049.610
## s.e.      809.0624      863.0465    817.8338  1955.179  1978.846
##             jan1      aug15      dec31     eastsun     oct31      nov1
##             4845.535 -7191.847 -3295.891 -22023.012 12584.843 13795.726
## s.e.      1989.392  2000.581  1991.405  1899.151  2001.358  2055.665
##             apr25      mag1      bridge Riscaldamento Pioggia_True
##             13020.889  5797.644  1878.9563 -33907.15  1304.9745
## s.e.      2130.635  1901.483  984.6478  11601.31   276.9706
##
## sigma^2 = 5232612: log likelihood = -3533.97
## AIC=7117.93  AICc=7121.53  BIC=7216.89
##
## Training set error measures:
##               ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set -25.9821 2215.425 1705.291 -2.040019 10.73664 0.2849353 0.00182366
```

```
pars_test(mod4_rid3$coef, mod4_rid3$var.coef)
```

```

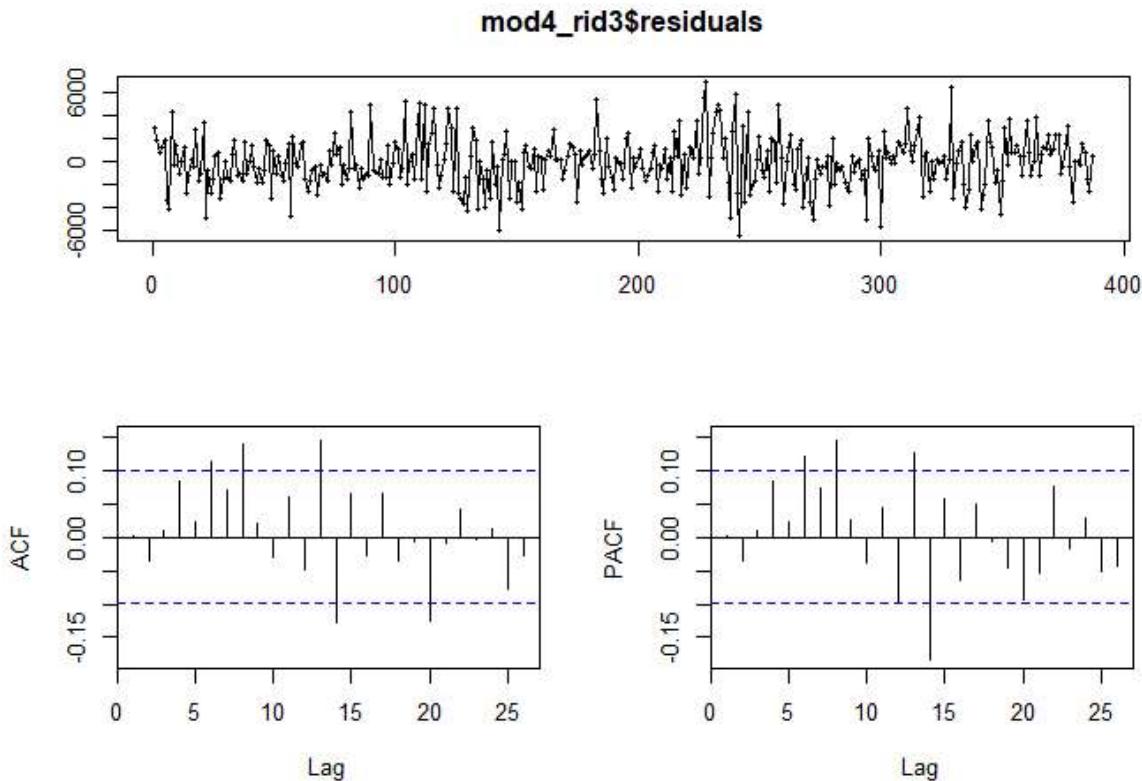
##          ar1          ar2          ar3          ma1          ma2
## 1.583508e-03 0.000000e+00 1.274758e-12 0.000000e+00 0.000000e+00
##          sar1          sma1      intercept Giorno_Friday Giorno_Saturday
## 0.000000e+00 8.018597e-06 1.783184e-04 1.953993e-14 0.000000e+00
## Giorno_Sunday      dec25       dec26        jan1        aug15
## 0.000000e+00 7.215993e-07 2.234591e-03 1.486355e-02 3.245429e-04
##          dec31      eastsun      oct31       nov1        apr25
## 9.791261e-02 0.000000e+00 3.212650e-10 1.931943e-11 9.884069e-10
##          mag1       bridge  Riscaldamento Pioggia_True
## 2.295962e-03 5.635864e-02 3.470109e-03 2.457812e-06

```

Ora tutte le variabili risultano essere statisticamente significative. Procediamo con la diagnostica del modello.

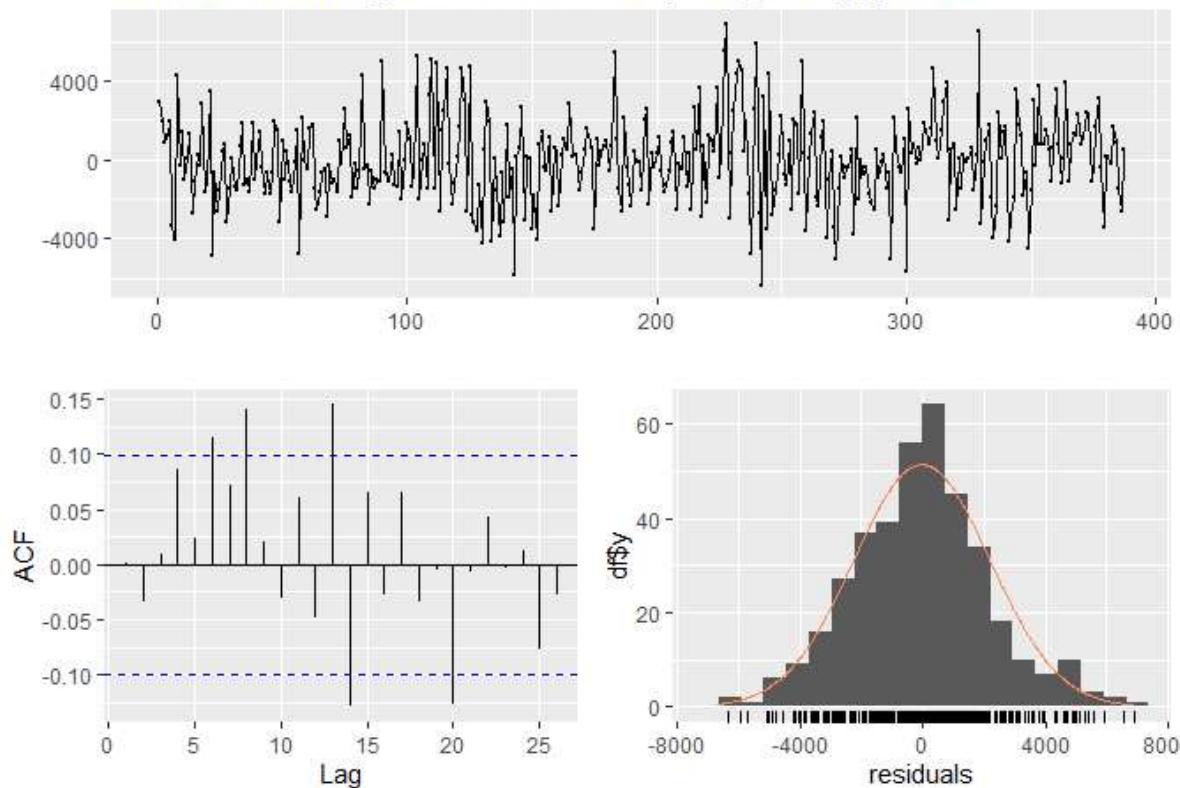
## Diagnostica

```
tsdisplay(mod4_rid3$residuals)
```



```
checkresiduals(mod4_rid3, test = FALSE)
```

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



```
checkresiduals(mod4_rid3, test = "LB", lag = 25, plot = FALSE)
```

```
##  
## Ljung-Box test  
##  
## data: Residuals from Regression with ARIMA(3,0,2)(1,0,1)[7] errors  
## Q* = 50.717, df = 18, p-value = 5.877e-05  
##  
## Model df: 7. Total lags used: 25
```

L'AIC risulta essere pari a : 7117.93

L'errore sul training set MAE risulta pari a 1705.291 mentre l'RMSE pari a: 2215.425

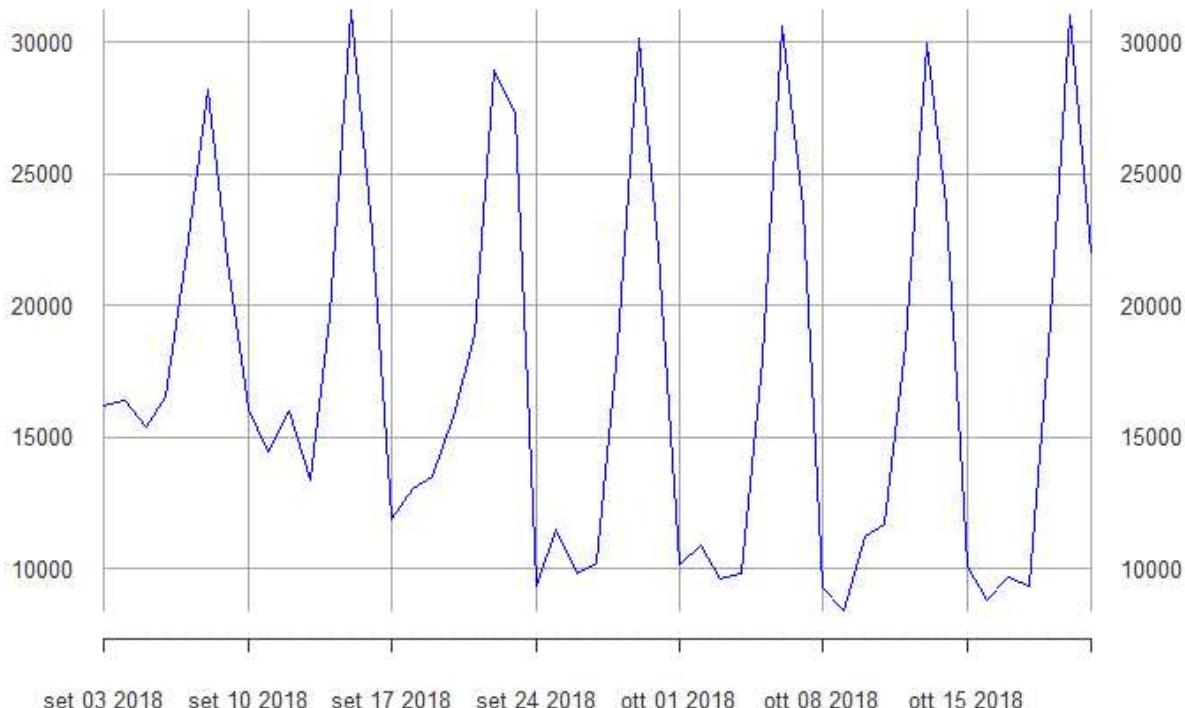
Il Box-Ljung test ( $p\text{-value} = 0.001$ ) ci indica che i residui presentano ancora auto-correlazione seriale.

Per quanto riguarda il fit del modello, possiamo notare come ci sia stato un netto miglioramento delle performance rispetto ai precedenti modelli. Permangono comunque ancora problemi di autocorrelazione, anche se osservando il grafico dell'ACF dei residui, le correlazioni che fuoriescono dalle bande di confidenza risultano comunque essere poche.

```
plot(train[1:49,1],  
      col = "blue", lwd=0.5,  
      main = "Fitted Values")
```

**Fitted Values**

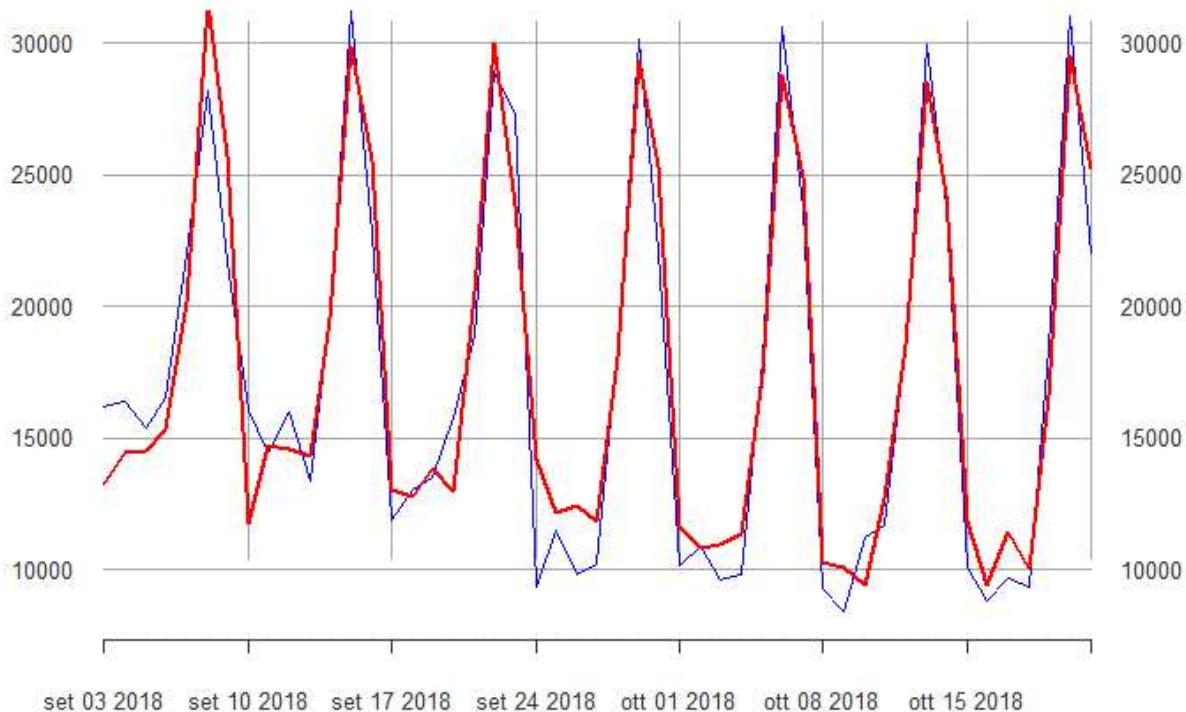
2018-09-03 / 2018-10-21



```
lines(xts(mod4_rid3$fitted[1:49], order.by = index(train[1:49])), col="red", lwd=2.5)
```

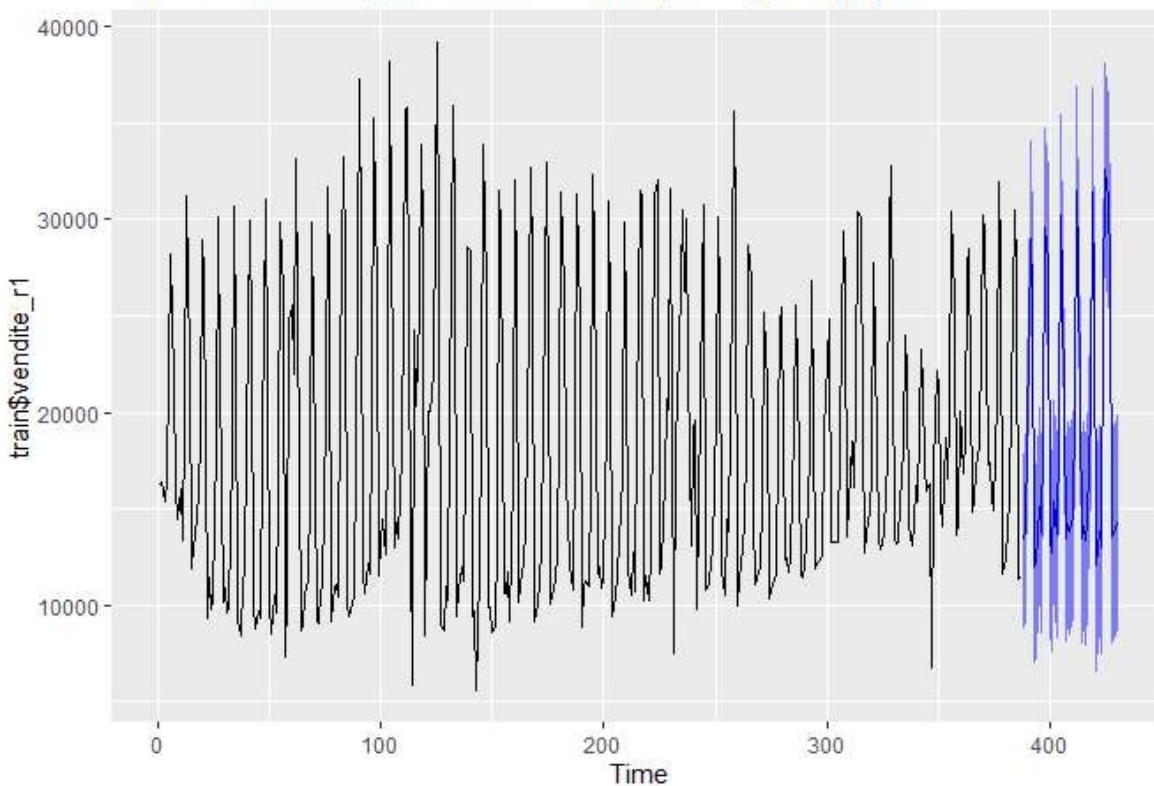
**Fitted Values**

2018-09-03 / 2018-10-21

**Previsioni**

```
# Al momento vengono fatte previsioni 44 passi in avanti (Lunghezza del validation set)
pred_mod4 <- forecast(mod4_rid3, h = 44,
                      level = 95,
                      xreg = validation[, -c(1,5,6,7,8,12,16,21,22)])
```

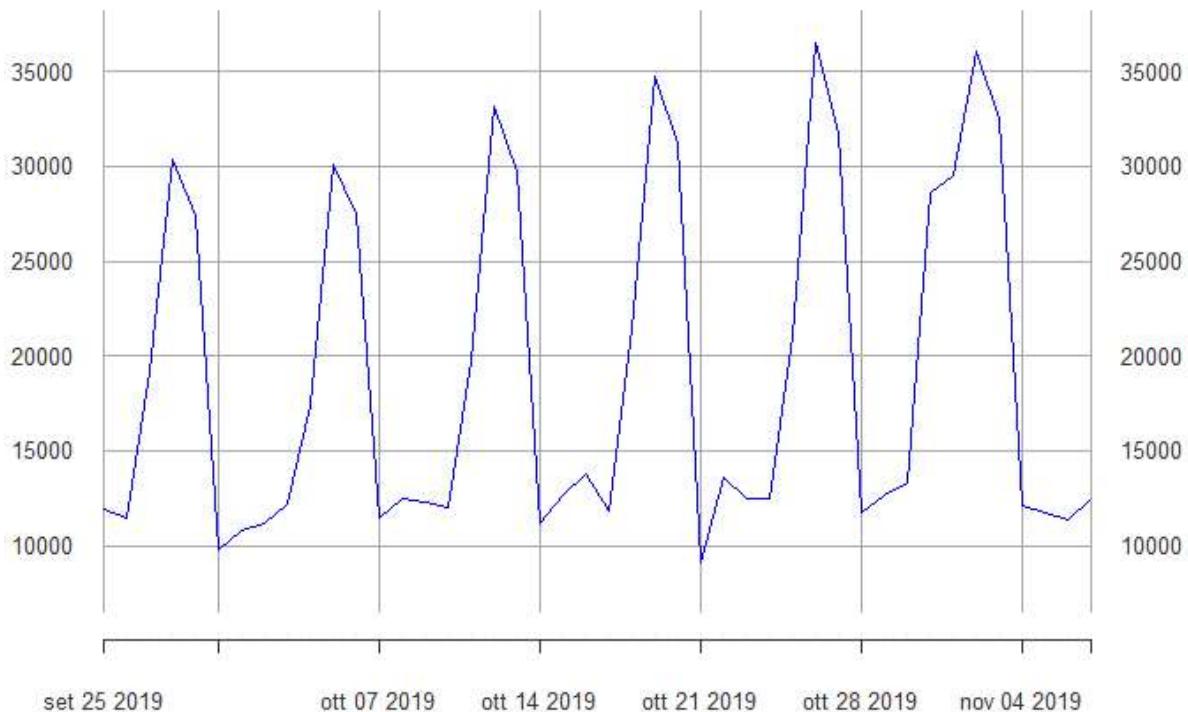
```
autoplot(pred_mod4)
```

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

```
plot(validation[,1],
  col = "blue", lwd=0.5,
  ylim = c(min(pred_mod4$lower, min(validation[,1])), max(pred_mod4$upper,
    max(validation[,1]))),
  main = "Fitted Value")
```

**Fitted Value**

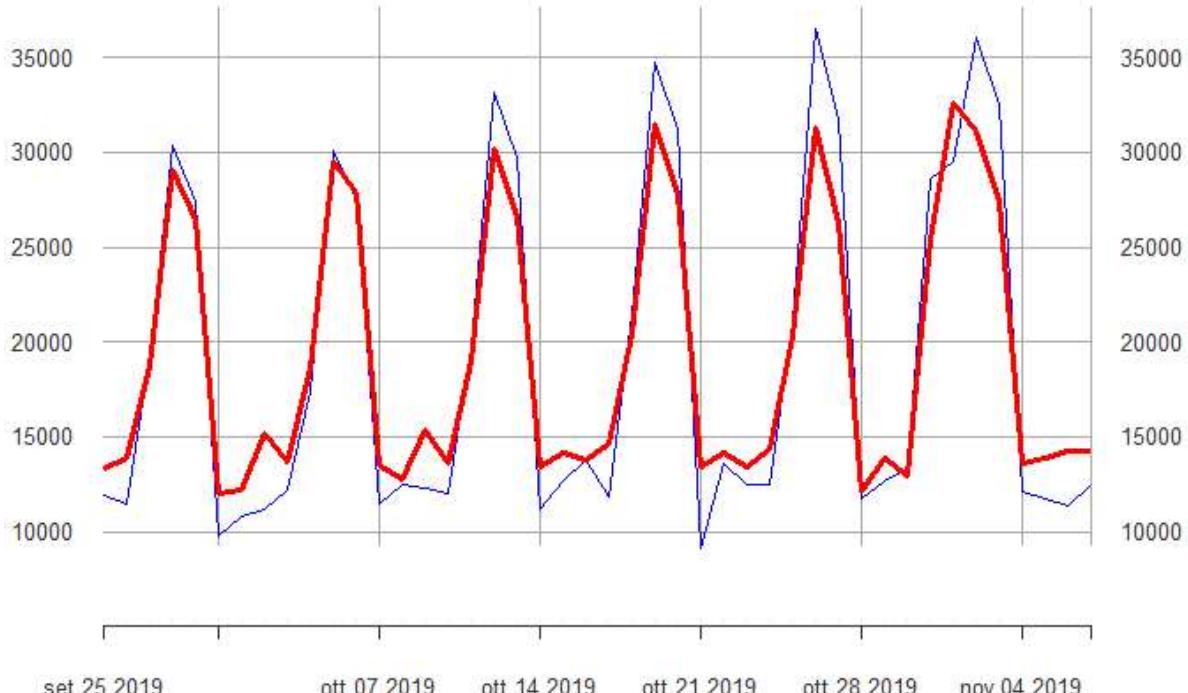
2019-09-25 / 2019-11-07



```
lines(xts(pred_mod4$mean, order.by = index(validation)), col="red", lwd=3)
```

**Fitted Value**

2019-09-25 / 2019-11-07



```
mape(validation[,1], xts(pred_mod4$mean, order.by = index(validation)))
```

```
## [1] 12.49453
```

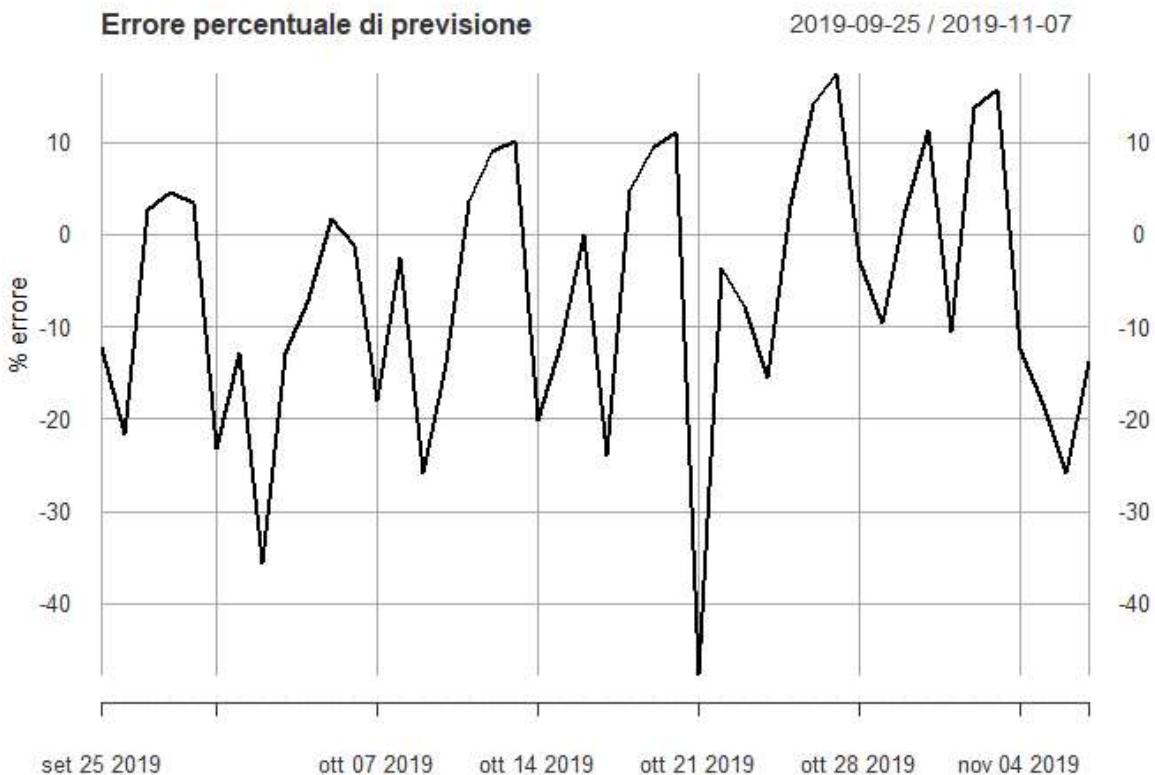
```
mae(validation[,1], xts(pred_mod4$mean, order.by = index(validation)))
```

```
## [1] 2091.466
```

```
rmse(validation[,1], xts(pred_mod4$mean, order.by = index(validation)))
```

```
## [1] 2547.985
```

```
err_plot(validation[,1], pred_mod4$mean)
```



Come precedentemente accennato, le previsioni anche sul validation set migliorano, attenstandosi su un MAE di 2091 rispetto ad un MAE di 2618 del modello3.

# 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
- Potranno 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

## Test su Mod1

```
# df per i modelli: 1,2,3
df<- read.zoo("../\\Dati aggiuntivi\\Dati CV_Arima\\df_123.csv", index.column = 1, sep =
      ",", header = TRUE)
df<- as.xts(df[,-c(5,6,7)])
head(df)
```

```
##           vendite_r1 Giorno_Friday Giorno_Saturday Giorno_Sunday
## 2018-09-03   16201.27          0            0            0
## 2018-09-04   16409.71          0            0            0
## 2018-09-05   15374.10          0            0            0
## 2018-09-06   16531.07          0            0            0
## 2018-09-07   22057.70          1            0            0
## 2018-09-08   28190.46          0            1            0
```

```
# df per il modello 4
df_x<- read.zoo("../\\Dati aggiuntivi\\Dati CV_Arima\\df_4.csv", index.column = 1, sep =
      ",", header = TRUE)
df_x<- as.xts(df_x[,-c(5,6,7,8,12,16,21,22)])
head(df_x)
```

```
##           vendite_r1 Giorno_Friday Giorno_Saturday Giorno_Sunday dec25 dec26
## 2018-09-03   16201.27          0            0            0            0            0
## 2018-09-04   16409.71          0            0            0            0            0
## 2018-09-05   15374.10          0            0            0            0            0
## 2018-09-06   16531.07          0            0            0            0            0
## 2018-09-07   22057.70          1            0            0            0            0
## 2018-09-08   28190.46          0            1            0            0            0
```

```

##                jan1 aug15 dec31 eastsun oct31 nov1 apr25 mag1 bridge Riscaldamento
## 2018-09-03      0      0      0      0      0      0      0      0      0      1.30325
## 2018-09-04      0      0      0      0      0      0      0      0      0      1.30325
## 2018-09-05      0      0      0      0      0      0      0      0      0      1.30325
## 2018-09-06      0      0      0      0      0      0      0      0      0      1.30325
## 2018-09-07      0      0      0      0      0      0      0      0      0      1.30325
## 2018-09-08      0      0      0      0      0      0      0      0      0      1.30325
##                Pioggia_True
## 2018-09-03      0
## 2018-09-04      0
## 2018-09-05      0
## 2018-09-06      1
## 2018-09-07      0
## 2018-09-08      0

```

```

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

```

```

## [1] "2022-12-06 12:20:50 CET"

```

# Definisco la forecast-function

```

f_mod1 <- function(x, h, xreg, newxreg) {
  forecast(Arima(x,
                  order = c(0, 0, 2),
                  seasonal = list(order = c(1, 0, 1), period = 7),
                  xreg = xreg,
                  include.constant = TRUE
                  )),
  h = h,
  xreg=newxreg)
}

xreg <- df[, -1]

initial = 90
e <- tscv(y = df$ vendite_r1, forecastfunction = f_mod1, h=90, xreg=xreg, initial = initial)
e <- tail(e, n = -initial)

```

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

```
## [1] "2022-12-06 12:21:32 CET"
```

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

```
## Time difference of 42.30764 secs
```

```
#print(e)  
#sqrt(mean(e^2, na.rm=TRUE))
```

```
print(nrow(df))
```

```
## [1] 539
```

```
print(nrow(e)+ initial)
```

```
## [1] 539
```

```
# Posso osservare dal print della coda della matrice degli errori come siano presenti NA,  
# questo conferma che tsCV() effettua CV con Non-Constant Holdout  
#print(tail(e))
```

```
#print(head(e, 5))  
print("MSE:")
```

```
## [1] "MSE:"
```

```
MSE_mod1 <- colMeans(e^2, na.rm = TRUE)
print(MSE_mod1)
```

```
##      h=1      h=2      h=3      h=4      h=5      h=6      h=7      h=8
## 11899807 13383264 13642613 13723893 13728460 13767417 13862722 14958446
##      h=9      h=10     h=11     h=12     h=13     h=14     h=15     h=16
## 15388414 15510082 15500430 15596823 15661327 15596594 15922409 16129966
##      h=17     h=18     h=19     h=20     h=21     h=22     h=23     h=24
## 16205418 16247024 16204642 16115607 16140492 16299117 16355917 16356866
##      h=25     h=26     h=27     h=28     h=29     h=30     h=31     h=32
## 16030946 15886231 15805507 15832716 15996625 16047794 15925880 15799843
##      h=33     h=34     h=35     h=36     h=37     h=38     h=39     h=40
## 15666953 15666713 15537774 15633047 15666738 15665657 15695695 15738048
##      h=41     h=42     h=43     h=44     h=45     h=46     h=47     h=48
## 15765425 15738833 15850654 15912959 15985284 16024446 16061826 16088765
##      h=49     h=50     h=51     h=52     h=53     h=54     h=55     h=56
## 16106709 16227962 16330874 16377606 16330847 16350088 16394610 16428644
##      h=57     h=58     h=59     h=60     h=61     h=62     h=63     h=64
## 16741269 16850373 16883711 16888662 16923556 16899805 16976142 17126835
##      h=65     h=66     h=67     h=68     h=69     h=70     h=71     h=72
## 17201932 17246982 17263811 17289563 17323313 17357690 17462112 17526510
##      h=73     h=74     h=75     h=76     h=77     h=78     h=79     h=80
## 17575217 17618052 17634644 17687594 17727752 17741289 17772841 17815176
##      h=81     h=82     h=83     h=84     h=85     h=86     h=87     h=88
## 17858867 17895584 17934227 17959525 18011839 18010708 18045588 18082085
##      h=89     h=90
## 18128954 18171470
```

```
print("RMSE:")
```

```
## [1] "RMSE:"
```

```
RMSE_mod1 <- sqrt(colMeans(e^2, na.rm = TRUE))
print(RMSE_mod1)
```

```
##      h=1      h=2      h=3      h=4      h=5      h=6      h=7      h=8
## 3449.610 3658.314 3693.591 3704.577 3705.194 3710.447 3723.268 3867.615
##      h=9      h=10     h=11     h=12     h=13     h=14     h=15     h=16
```

```

## 3922.807 3938.284 3937.059 3949.281 3957.440 3949.252 3990.289 4016.213
## h=17     h=18     h=19     h=20     h=21     h=22     h=23     h=24
## 4025.595 4030.760 4025.499 4014.425 4017.523 4037.216 4044.245 4044.362
## h=25     h=26     h=27     h=28     h=29     h=30     h=31     h=32
## 4003.866 3985.754 3975.614 3979.035 3999.578 4005.970 3990.724 3974.902
## h=33     h=34     h=35     h=36     h=37     h=38     h=39     h=40
## 3958.150 3958.120 3941.798 3953.865 3958.123 3957.987 3961.779 3967.121
## h=41     h=42     h=43     h=44     h=45     h=46     h=47     h=48
## 3970.570 3967.220 3981.288 3989.105 3998.160 4003.055 4007.721 4011.080
## h=49     h=50     h=51     h=52     h=53     h=54     h=55     h=56
## 4013.316 4028.394 4041.148 4046.926 4041.144 4043.524 4049.026 4053.226
## h=57     h=58     h=59     h=60     h=61     h=62     h=63     h=64
## 4091.610 4104.921 4108.979 4109.582 4113.825 4110.937 4120.211 4138.458
## h=65     h=66     h=67     h=68     h=69     h=70     h=71     h=72
## 4147.521 4152.949 4154.974 4158.072 4162.128 4166.256 4178.769 4186.468
## h=73     h=74     h=75     h=76     h=77     h=78     h=79     h=80
## 4192.281 4197.386 4199.362 4205.662 4210.434 4212.041 4215.785 4220.803
## h=81     h=82     h=83     h=84     h=85     h=86     h=87     h=88
## 4225.975 4230.317 4234.882 4237.868 4244.036 4243.902 4248.010 4252.304
## h=89     h=90
## 4257.811 4262.801

```

```
print("MAE:")
```

```
## [1] "MAE:"
```

```
MAE_mod1 <- colMeans(abs(e), na.rm = TRUE)
print(MAE_mod1)
```

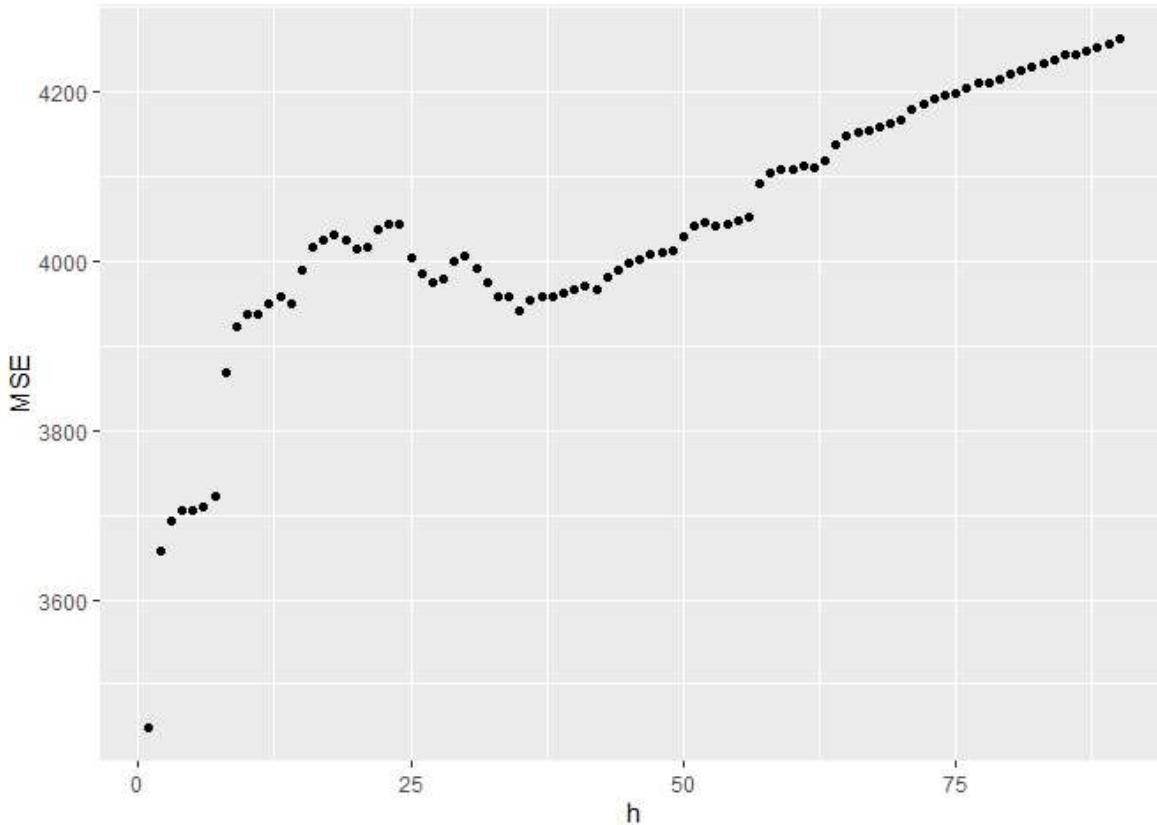
```

## h=1     h=2     h=3     h=4     h=5     h=6     h=7     h=8
## 2260.978 2478.392 2507.805 2521.032 2527.417 2528.917 2536.767 2681.450
## h=9     h=10    h=11    h=12    h=13    h=14    h=15    h=16
## 2723.219 2742.070 2739.223 2751.182 2755.728 2745.966 2776.209 2803.079
## h=17    h=18    h=19    h=20    h=21    h=22    h=23    h=24
## 2810.042 2813.786 2808.903 2799.162 2795.219 2814.867 2823.878 2822.334
## h=25    h=26    h=27    h=28    h=29    h=30    h=31    h=32
## 2801.147 2785.483 2772.080 2769.357 2784.979 2786.043 2776.358 2764.196
## h=33    h=34    h=35    h=36    h=37    h=38    h=39    h=40
## 2751.544 2746.719 2731.495 2742.334 2736.607 2736.211 2739.086 2743.426
## h=41    h=42    h=43    h=44    h=45    h=46    h=47    h=48

```

```
## 2745.794 2737.794 2739.955 2744.029 2754.711 2759.570 2765.502 2768.094
## h=49      h=50      h=51      h=52      h=53      h=54      h=55      h=56
## 2764.972 2771.486 2779.402 2779.479 2770.116 2775.803 2777.200 2772.377
## h=57      h=58      h=59      h=60      h=61      h=62      h=63      h=64
## 2787.405 2793.426 2797.540 2794.420 2793.713 2784.873 2791.471 2806.246
## h=65      h=66      h=67      h=68      h=69      h=70      h=71      h=72
## 2813.322 2816.589 2819.722 2819.545 2818.613 2825.793 2833.909 2835.632
## h=73      h=74      h=75      h=76      h=77      h=78      h=79      h=80
## 2841.264 2848.335 2847.806 2856.523 2858.306 2855.735 2855.736 2856.719
## h=81      h=82      h=83      h=84      h=85      h=86      h=87      h=88
## 2862.146 2862.956 2863.678 2863.776 2869.630 2870.110 2874.508 2879.125
## h=89      h=90
## 2881.659 2885.965
```

```
data.frame(h = 1:90, MSE = RMSE_mod1) %>%
  ggplot(aes(x = h, y = MSE)) + geom_point()
```



Ora stimo il modello nella sua interezza e confronto RMSE ottenuto dai residui con quello ottenuto con CV:

```
mod1 <- Arima(y = df$vendite_r1,
                 order = c(0, 0, 2),
```

```

list(order = c(1, 0, 1), period = 7),
xreg = df[, -1],
include.constant = TRUE,
)
summary(mod1)

```

```

## Series: df$vendite_r1
## Regression with ARIMA(0,0,2)(1,0,1)[7] errors
##
## Coefficients:
##             ma1      ma2     sar1     sma1   intercept Giorno_Friday
##             0.3887  0.1687  0.7981 -0.5869  12912.7345    6572.5598
## s.e.      0.0446  0.0426  0.0842  0.1204   509.2502    798.2441
##             Giorno_Saturday Giorno_Sunday
##                 18428.1881    15003.1727
## s.e.        865.4274     800.6698
##
## sigma^2 = 10508386: log likelihood = -5118.66
## AIC=10255.31  AICc=10255.65  BIC=10293.92
##
## Training set error measures:
##                  ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -5.474709 3217.517 2113.583 -3.842661 13.28754 0.3377006
##                  ACF1
## Training set 0.008652757

```

- Come atteso l'RMSE in previsione è sempre maggiore di quello ottenuto sui residui di trainin (medi primo allenamento del modello)

## Confronto tra i modelli

### Cross-Validation

```

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

```

```

## [1] "2022-12-06 12:21:33 CET"

```

```
# Definisco la forecast-function per i diversi modelli

#1_ARIMA
f_mod1 <- function(x, h, xreg, newxreg) {
  forecast(Arima(x,
    order = c(0, 0, 2),
    seasonal = list(order = c(1, 0, 1), period = 7),
    xreg = xreg,
    include.constant = FALSE
  ),
  h = h,
  xreg=newxreg)
}

#2_ARIMA
f_mod2 <- function(x, h, xreg, newxreg) {
  forecast(Arima(x,
    order = c(5, 0, 2),
    seasonal = list(order = c(1, 0, 1), period = 7),
    xreg = xreg,
    include.constant = TRUE
  ),
  h = h,
  xreg=newxreg)
}

#3_ARIMA
f_mod3 <- function(x, h, xreg, newxreg) {
  forecast(Arima(x,
    order = c(3, 0, 2),
    seasonal = list(order = c(1, 0, 1), period = 7),
    xreg = xreg,
    include.constant = TRUE
  ),
  h = h,
  xreg=newxreg)
}

#4_SARIMAX
f_mod4 <- function(x, h, xreg, newxreg) {
  forecast(Arima(x,
    order = c(3, 0, 2),
    seasonal = list(order = c(1, 0, 1), period = 7),
    xreg = xreg,
    include.constant = TRUE
  ),
}
```

```

    h = h,
    xreg=newxreg)
}

# Parametri di cross-validation globali
h = 90
initial = 60

# Stima degli errori di previsione con CV

xreg1 <- df[, -1]
e1 <- tsCV(y = df$vendite_r1, forecastfunction = f_mod1, h=h, xreg=xreg1, initial =
    initial)
e1 <- tail(e1, n = -initial)

xreg2 <- df[, -1]
e2 <- tsCV(y = df$vendite_r1, forecastfunction = f_mod2, h=h, xreg=xreg2, initial =
    initial)
e2 <- tail(e2, n = -initial)

xreg3 <- df[, -1]
e3 <- tsCV(y = df$vendite_r1, forecastfunction = f_mod3, h=h, xreg=xreg3, initial =
    initial)
e3 <- tail(e3, n = -initial)

# Runna il preprocessing nella sezione del modello 4 (df_reg) !
xreg4 <- df_x[, -1]
e4 <- tsCV(y = df_x$vendite_r1, forecastfunction = f_mod4, h=h, xreg=xreg4, initial =
    initial)
e4 <- tail(e4, n = -initial)

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

```

```
## [1] "2022-12-06 12:36:53 CET"
```

```

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

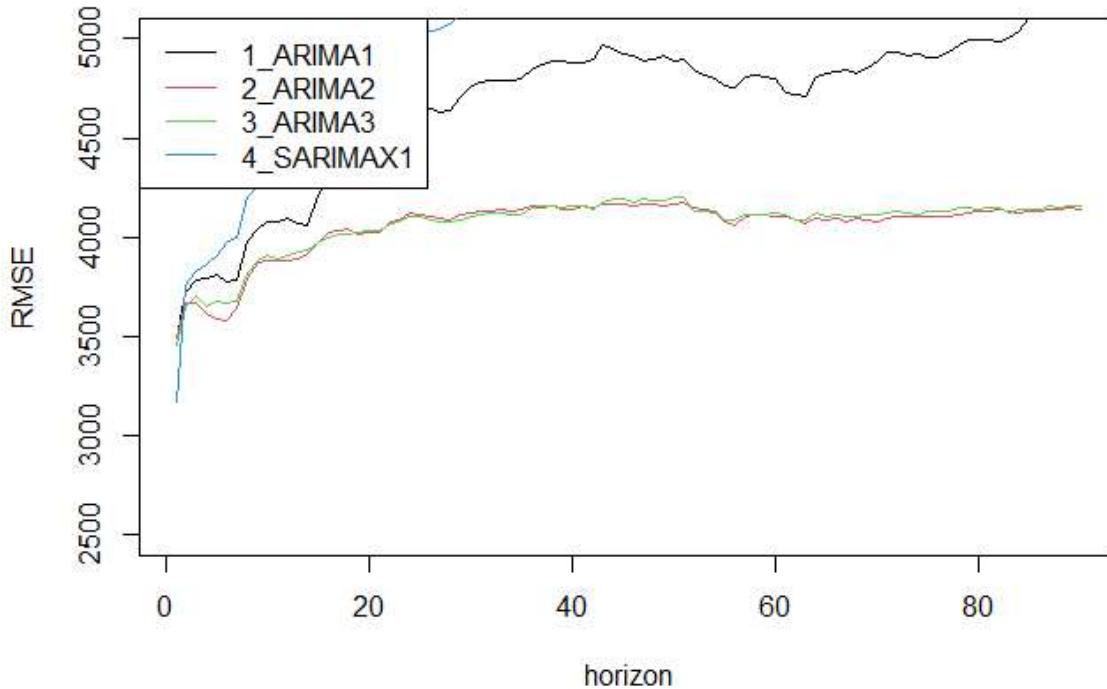
```

```
## Time difference of 15.33309 mins
```

## Accuratezze

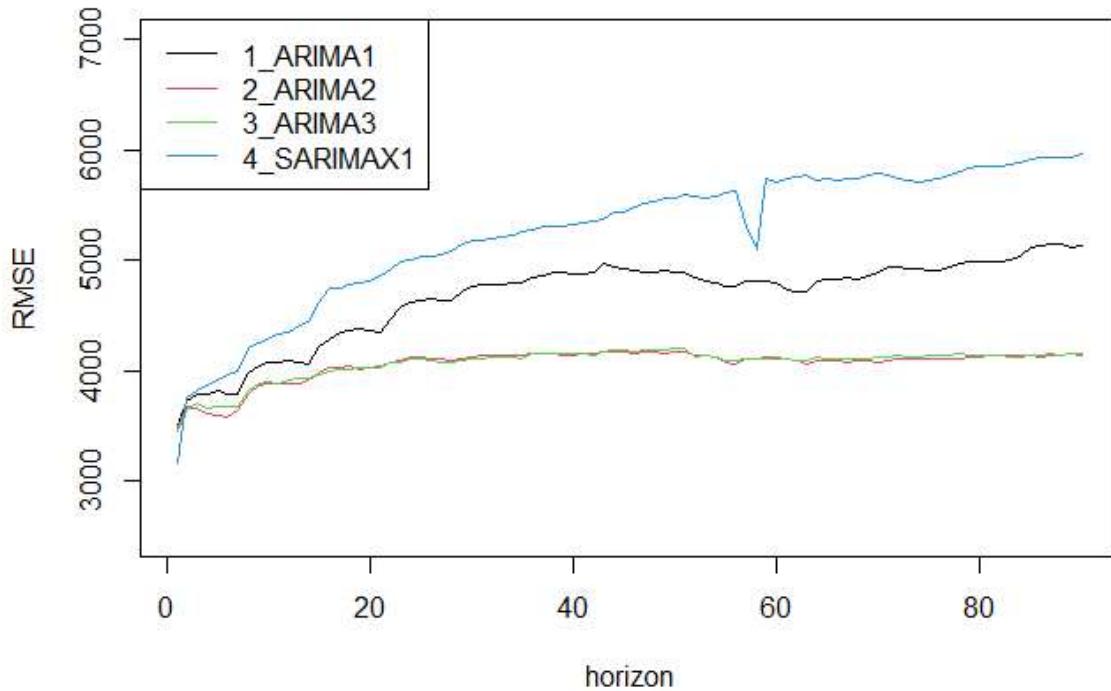
```
RMSE_mod1 <- sqrt(colMeans(e1^2, na.rm = TRUE))
RMSE_mod2 <- sqrt(colMeans(e2^2, na.rm = TRUE))
RMSE_mod3 <- sqrt(colMeans(e3^2, na.rm = TRUE))
RMSE_mod4 <- sqrt(colMeans(e4^2, na.rm = TRUE))
#RMSE_mod6 <- sqrt(colMeans(e6^2, na.rm = TRUE))

# Zoom in
plot(1:90, RMSE_mod1, type="l", col=1, xlab="horizon", ylab="RMSE", ylim = c(2500,5000))
lines(1:90, RMSE_mod2, type="l", col=2)
lines(1:90, RMSE_mod3, type="l", col=3)
lines(1:90, RMSE_mod4, type="l", col=4)
legend("topleft", legend=c("1_ARIMA1", "2_ARIMA2", "3_ARIMA3", "4_SARIMAX1"), col=1:4, lty=1)
```



```
# Zoom out
plot(1:90, RMSE_mod1, type="l", col=1, xlab="horizon", ylab="RMSE", ylim = c(2500,7000))
lines(1:90, RMSE_mod2, type="l", col=2)
```

```
lines(1:90, RMSE_mod3, type="l", col=3)
lines(1:90, RMSE_mod4, type="l", col=4)
legend("topleft", legend=c("1_ARIMA1", "2_ARIMA2", "3_ARIMA3", "4_SARIMAX1"), col=1:4, lty=1)
```



## RMSE Medi

```
mean(RMSE_mod1)
```

```
## [1] 4671.31
```

```
mean(RMSE_mod2)
```

```
## [1] 4055.611
```

```
mean(RMSE_mod3)
```

```
## [1] 4065.781
```

```
mean(RMSE_mod4)
```

```
## [1] 5250.578
```