

Pranav Kasela 846965

```
library(ggplot2)
library(forecast)
library(tidyverse)

df <- read.csv2("time_series_dataset.csv", dec = ".")
df$Data <- as.Date(df$Data)
head(df)
```

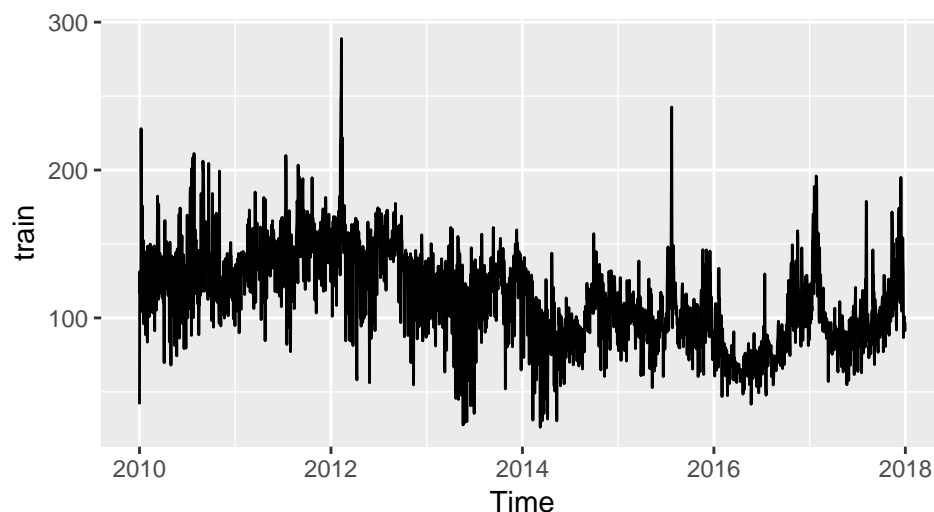
```
##      Data      value
## 1 2010-01-01 41.65104
## 2 2010-01-02 131.28660
## 3 2010-01-03 117.38812
## 4 2010-01-04 116.46128
## 5 2010-01-05 123.82376
## 6 2010-01-06 104.28556
```

Arima Models

Splittiamo il dataset in train e test per verificare come va il modello su nuovi dati.

```
idata <- ts(df$value, start=c(2010,1), frequency=365)
train <- window(idata, start=c(2010,1), end=c(2017,365))
#train <- df[1:(nrow(df) - 365),] #last one year is for validation
test <- window(idata, start=c(2018,1), end=c(2018,365))
#test <- df[(nrow(df) - 365 + 1):nrow(df),]

autoplot(train)
```



```
#This is the function ggtsdisplay of forecast package,
#but it has been modified so it doesn't plot the series,
#just the ACF and PACF plot, with the horizontal parameter
#the plot can be either horizontal or vertical
```

```

#The function has been simplified a lot, since we don't need
#all the complexity the original one has.
gtsdisplay_2 <- function(x, lag.max, horizontal=TRUE, ...) {
  if (!is.ts(x)) {
    x <- ts(x)
  }
  if (missing(lag.max)) {
    lag.max <- round(min(max(10 * log10(length(x)), 3 * frequency(x)), length(x) / 3))
  }
  #####      END      CHECKING      #####

  # Set up grid for plots
  if (horizontal){
    gridlayout <- matrix(c(2, 3), nrow = 1)
  }
  else{
    gridlayout <- matrix(c(2, 3), nrow = 2)
  }
  grid::grid.newpage()
  grid::pushViewport(grid::viewport(layout = grid::grid.layout(nrow(gridlayout), ncol(gridlayout))))

  # Prepare Acf plot
  acfplot <- do.call(ggAcf, c(x = quote(x), lag.max = lag.max)) +
    ggplot2::ggtitle("ACF") + ggplot2::ylab(NULL)

  # Prepare last plot (variable)
  pacfplot <- ggPacf(x, lag.max = lag.max) + ggplot2::ggtitle("PACF") +
    ggplot2::ylab(NULL)
  # Match y-axis
  acfplotrange <- ggplot2::layer_scales(acfplot)$y$range$range
  pacfplotrange <- ggplot2::layer_scales(pacfplot)$y$range$range
  yrange <- range(c(acfplotrange, pacfplotrange))
  acfplot <- acfplot + ggplot2::ylim(yrange)
  pacfplot <- pacfplot + ggplot2::ylim(yrange)

  # Add ACF plot
  matchidx <- as.data.frame(which(gridlayout == 2, arr.ind = TRUE))
  print(
    acfplot,
    vp = grid::viewport(
      layout.pos.row = matchidx$row,
      layout.pos.col = matchidx$col
    )
  )

  # Add PACF plot
  matchidx <- as.data.frame(which(gridlayout == 3, arr.ind = TRUE))
  print(
    pacfplot,
    vp = grid::viewport(
      layout.pos.row = matchidx$row,
      layout.pos.col = matchidx$col
    )
  )
}

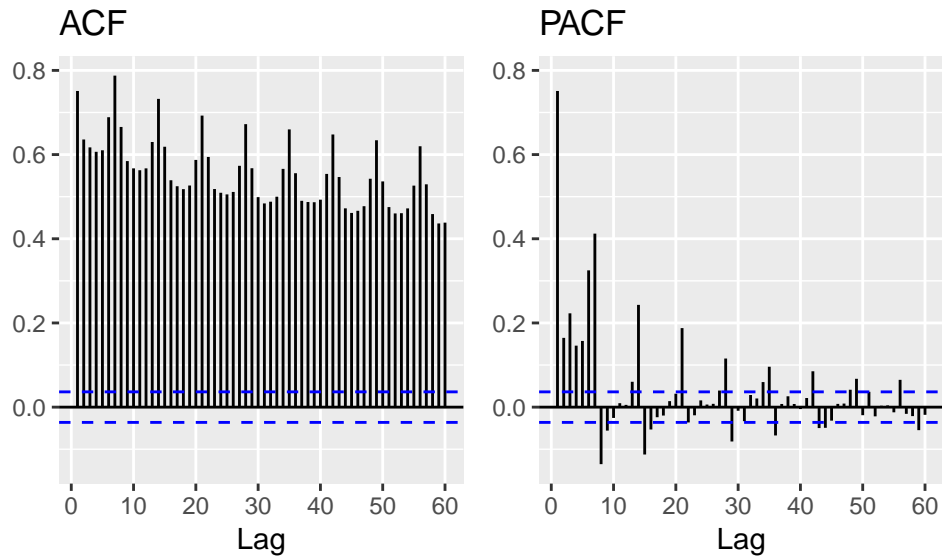
```

```

)
}

ggtsdisplay_2(train, horizontal = TRUE, lag.max = 60)

```

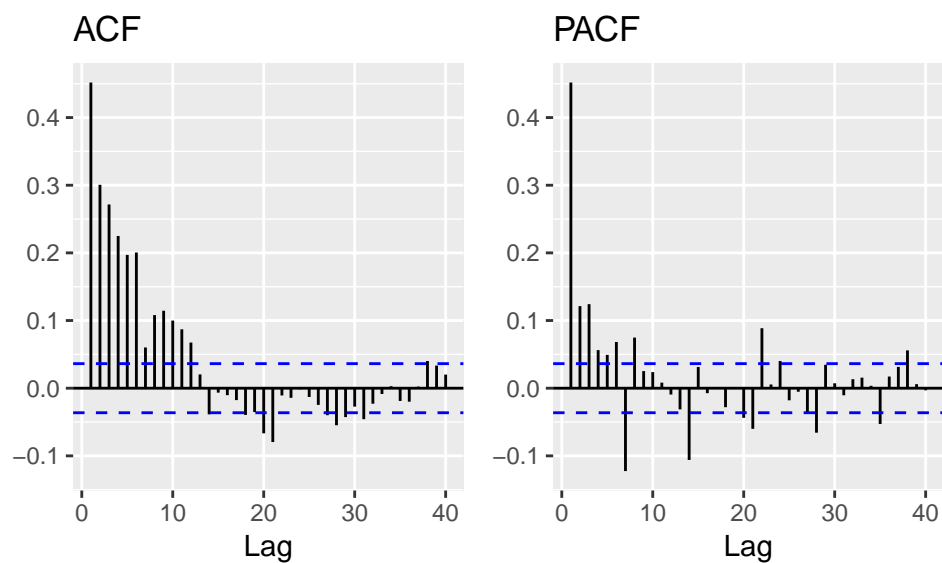


Possiamo vedere che nel PACF vi sono 7 ritardi e il ritratto stagionale che scende esponenzialmente al settimo ritardo, indicando la presenza di un $SMA(1)_7$ e vedendo ACF dai primi 2 ritardi stagionali ci convinciamo dell'esistenza di $SMA(1)_7$, inoltre vi è presente anche un $SAR(1)_7$. Stagionalità 7 indica un periodo settimanale in questa serie. Inoltre vista la discesa lenta e non geometrica della ACF potrebbe suggerire l'esistenza di una integrazione stagionale. Iniziamo ad aggiungere la parte stagionale prima e cerchiamo di capire dai residui come andrebbe aggiustato il modello.

```

mod1 <- Arima(train, c(0,0,0), list(order=c(1,0,1), period=7), lambda = "auto")
ggtsdisplay_2(mod1$residuals, lag.max = 40)

```



```
mod1
```

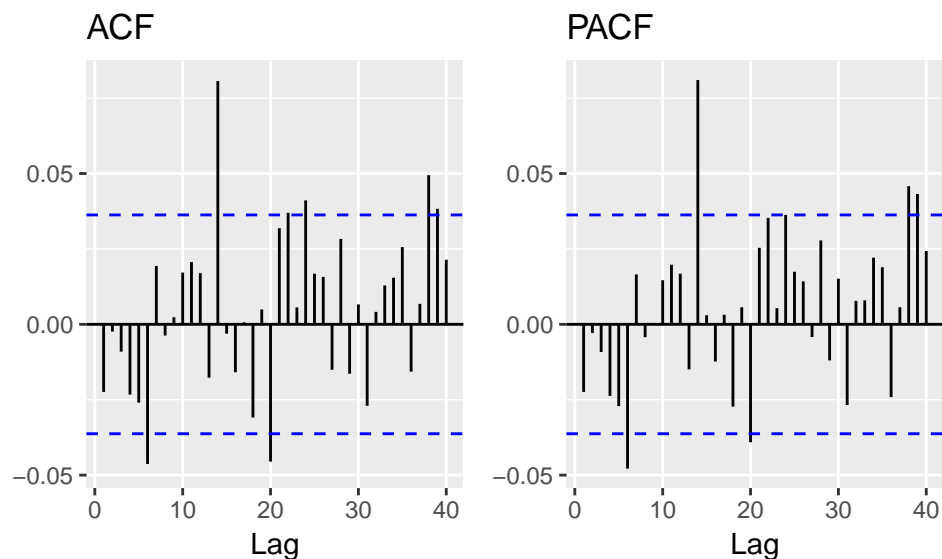
```
## Series: train
## ARIMA(0,0,0)(1,0,1)[7] with non-zero mean
## Box Cox transformation: lambda= 0.8027839
##
## Coefficients:
##          sar1          sma1          mean
##          0.9596      -0.5540      54.9480
## s.e.    0.0062      0.0224      1.3391
##
## sigma^2 estimated as 47.54:  log likelihood=-9784.64
## AIC=19577.28  AICc=19577.29  BIC=19601.2
```

Vediamo anche che il coefficiente di SAR è molto vicino ad 1, quindi ha radice unitaria e ciò dice che esiste l'integrazione stagionale che sospettavamo prima.

```
mod2 <- Arima(train, c(6,0,0), list(order=c(1,1,1), period=7), lambda = "auto")
mod2
```

```
## Series: train
## ARIMA(6,0,0)(1,1,1)[7]
## Box Cox transformation: lambda= 0.8027839
##
## Coefficients:
##          ar1          ar2          ar3          ar4          ar5          ar6          sar1          sma1
##          0.4136      0.0663      0.1164      0.0634      0.0439      0.1398      0.0834     -0.9545
## s.e.    0.0186      0.0202      0.0201      0.0201      0.0201      0.0192      0.0215      0.0090
##
## sigma^2 estimated as 33.1:  log likelihood=-9233.68
## AIC=18485.36  AICc=18485.42  BIC=18539.15
```

```
ggttsdisplay_2(mod2$residuals, lag.max = 40)
```



I residui sembrano essere rientrati nella banda tranne un residuo a 14 sia un ACF che PACF.

Usiamo il test Augmented Dickey-Fuller, che cerca radici unitarie nella serie, per verificare se la serie è stazionaria, con k che indica il ritardo autoregressivo, H_0 è la presenza di radici unitarie nella serie, H_1 è che

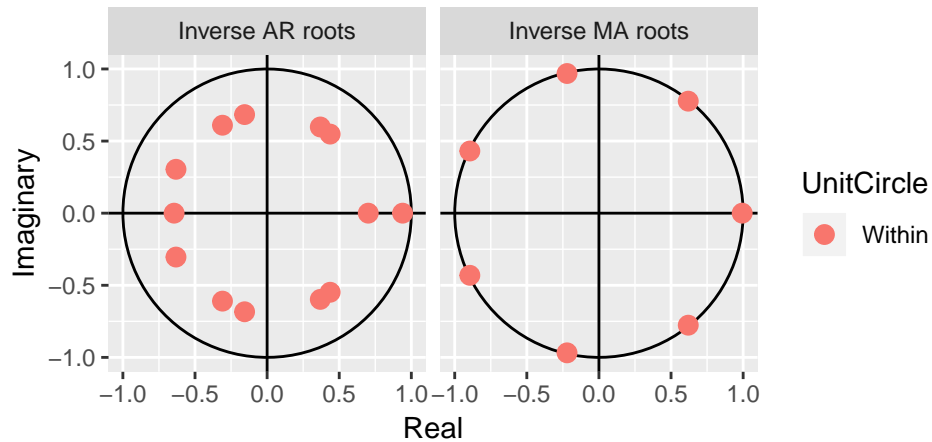
la serie è stazionaria.

```
#Trying Augmented Dickey-Fuller test to see if the series is stationary:  
##H_0$ is that the model is not stationary  
tseries::adf.test(mod2$residuals, k=7)
```

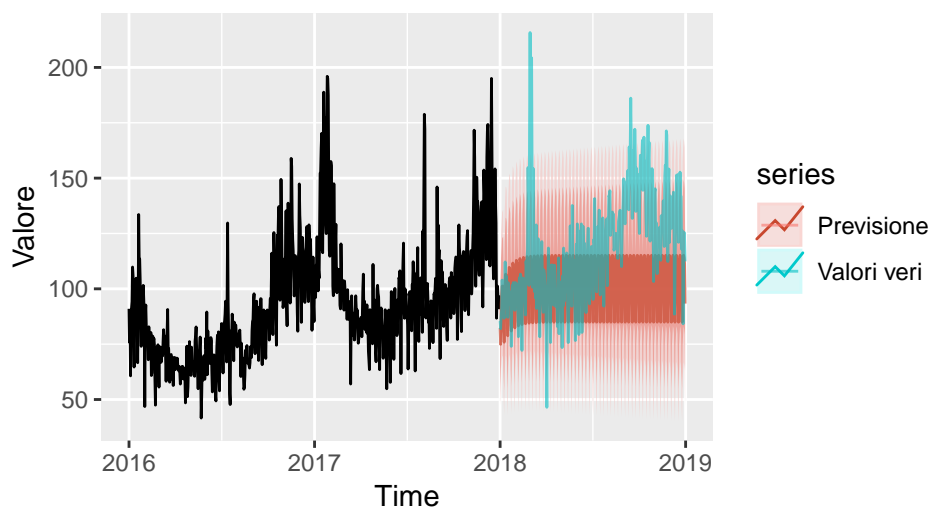
```
##  
## Augmented Dickey-Fuller Test  
##  
## data: mod2$residuals  
## Dickey-Fuller = -20.344, Lag order = 7, p-value = 0.01  
## alternative hypothesis: stationary
```

Poiché il p-value è basso rifiutiamo l' H_0 e possiamo dire che i residui della sono stazionari. Questo è per il modello AR, mentre per SAR lo abbiamo già visto prima.

```
#Another way, it shows the roots  
autoplot(mod2)
```



```
pred <- forecast(mod2, h=365)  
  
autoplot(window(idata, start=c(2016,1), end=c(2017,365))) +  
  autolayer(pred, ts.colour="black",series="Previsione", alpha=0.7) +  
  autolayer(test, series="Valori veri", alpha=0.6) +  
  xlab("Time") +  
  ylab("Valore")
```

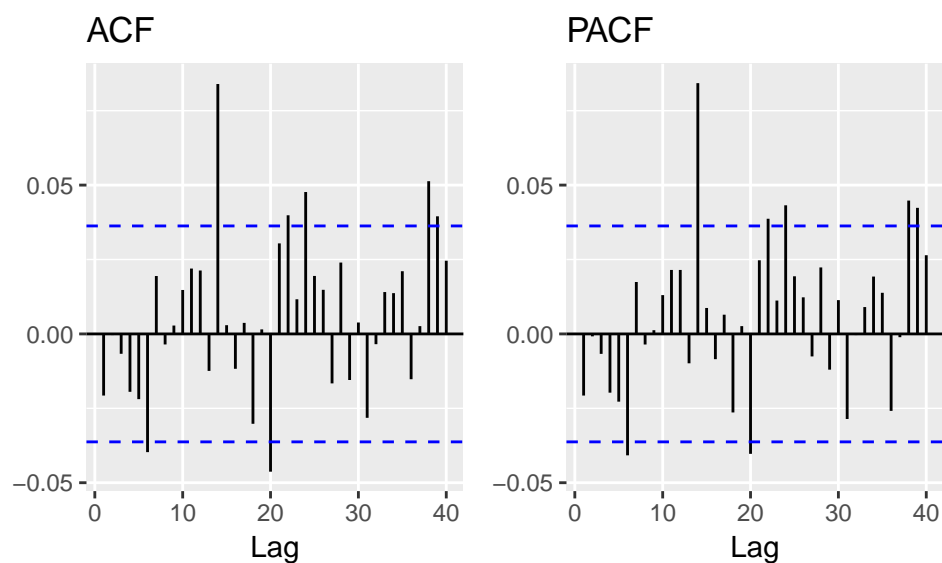


Mettiamo regressori dummy mensili e ogni 4 settimane.

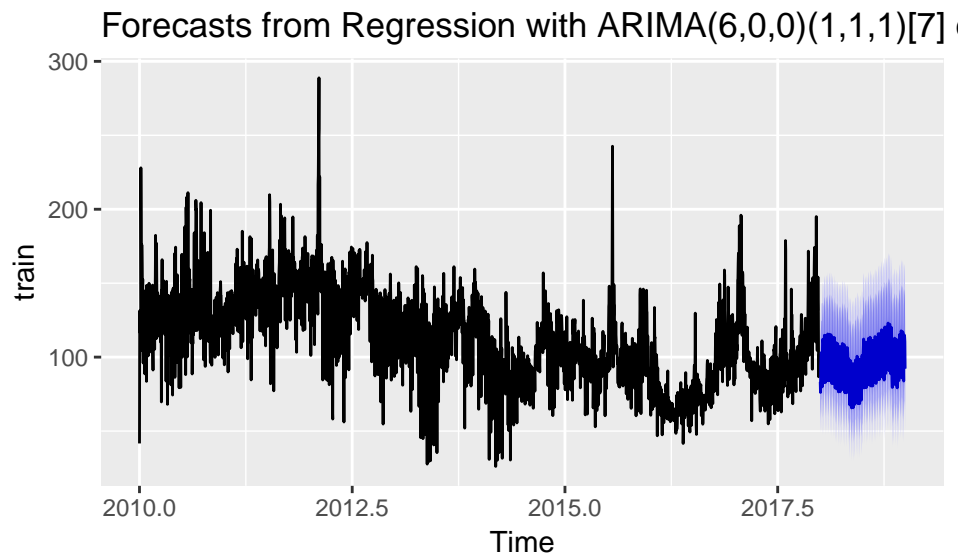
```
#create dummy
data.frame(Data=df$Data) %>%
  mutate(M = months(Data), ind = 1) %>%
  spread(M, ind, fill = 0) %>%
  mutate(W = paste0("W", (data.table::week(Data) %% 4 + 1)), ind=1) %>%
  spread(W, ind, fill = 0) %>%
  select(-starts_with("Data")) %>% as.matrix() -> more_reg

#eliminiamo la prima e l'ultima colonna, per evitare multicollinearità
xreg <- more_reg[,2:(ncol(more_reg)-1)]

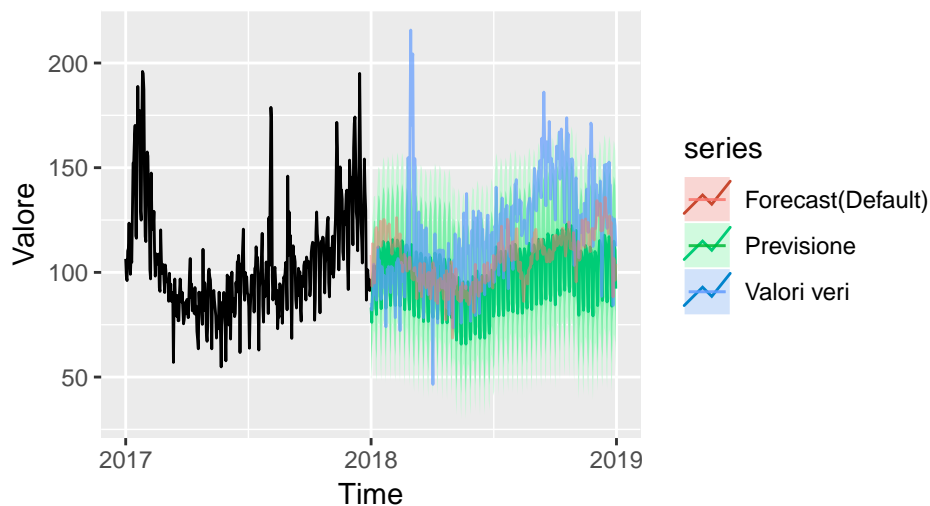
mod1_reg <- Arima(train, c(6,0,0), list(order=c(1,1,1), period=7),
  xreg=xreg[1:(length(train)),], include.constant = TRUE, lambda = "auto")
ggtsdisplay_2(mod1_reg$residuals, lag.max = 40)
```



```
pred_reg <- forecast(mod1_reg, h=365,
                     xreg=xreg[(length(train)+1):(length(train)+365),])
autoplot(pred_reg)
```



```
autoplot(window(idata, start=c(2017,1), end=c(2017,365))) +
  autolayer(pred_reg, series="Previsione") +
  autolayer(test, series="Valori veri", alpha=0.7) +
  autolayer(forecast(train, h=365)$mean, alpha=0.5, series="Forecast(Default)") +
  ylab("Valore")
```



Questa serie presenta in realtà una multi-stagionalità, che non può essere risolta con R, per questo si prova ad usare regressori esterni, introducendo una stagionalità annuale e ogni 4 settimane.

```
mod_reg_y <- forecast(mod1_reg, h=365,
                     xreg=xreg[(nrow(df)-365+1):(length(train)+365),])$mean
mod2_y <- forecast(mod2, h=365)$mean
```

```

mse_reg <- sqrt(mean((test - mod_reg_y)^2))
mse_2 <- sqrt(mean((test - mod2_y)^2))

print(paste0("MSE relativo modello senza regressori : ", mse_2))

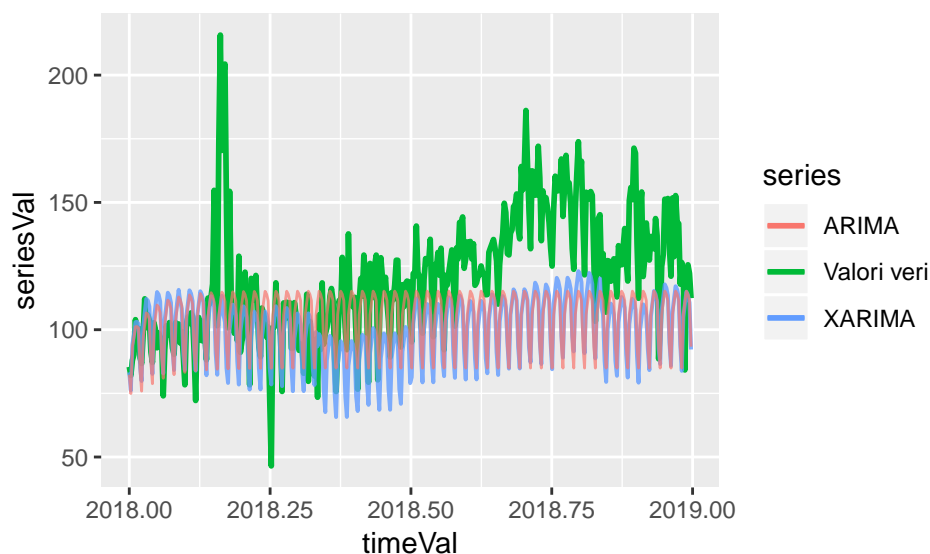
## [1] "MSE relativo modello senza regressori : 27.2573932289143"

print(paste0("MSE relativo modello con regressori : ", mse_reg))

## [1] "MSE relativo modello con regressori : 28.7326528499392"

ggplot() +
  autolayer(test, series="Valori veri", size=1) +
  autolayer(pred_reg$mean, series="XARIMA", size=0.7, alpha=0.8) +
  autolayer(pred$mean, series="ARIMA", alpha=0.7)

```



```
#autolayer(forecast(train, h=365)$mean, alpha=0.7, series="Forecast") +
```

In realtà si nota che il modello performa peggio sul validation se vengono fornite regressori esterni per indicare il mese e la settimana, ciò è dovuto al fatto che nell'ultimo anno l'andamento è diverso dagli ultimi anni, oppure queste variabili non sono tanto esplicative quanto si credeva e aggiungono solo del rumore alla previsione.

UCM

```

library(KFAS)

ytrain <- as.numeric(train)
ytrain[(length(ytrain)+1):(length(ytrain)+365)] <- NA

mod1 <- SSMModel(ytrain ~ 0 +
  SSMtrend(1, NA) +
  SSMseasonal(7, NA, "dummy") +
  SSMseasonal(365, 0, "trig",
    harmonics = 1:12),
  H = NA)

```



```

vary <- var(ytrain, na.rm = TRUE)
mod1$P1inf <- mod1$P1inf * 0
mod1$a1[1] <- mean(ytrain, na.rm = TRUE)
diag(mod1$P1) <- vary

# Initial values for the variances we have to estimate
init <- numeric(3)
init[1] <- log(vary/10) # log-var(dist.rw)
init[2] <- log(vary/100) # log-var(dist.seas)
init[3] <- log(vary/10) # log-var(err.oss.)

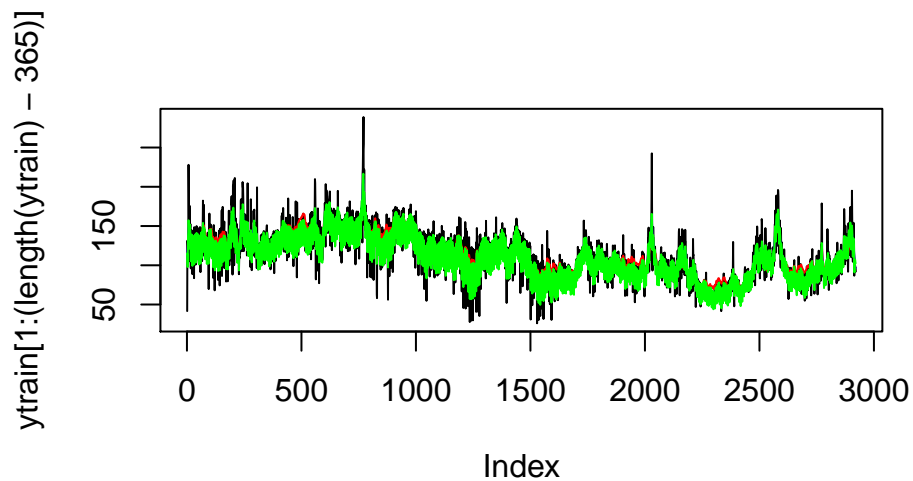
# Estimate
fit1 <- fitSSM(mod1, init)
fit1$optim.out$convergence

## [1] 0

smo1 <- KFS(fit1$model, smoothing = c("state", "disturbance", "signal"))
smo1_seas <- rowSums(smo1$alphahat[1:(length(ytrain) - 365), seq(8, 27, 2)])

plot(ytrain[1:(length(ytrain) - 365)], type = "l")
lines(smo1$alphahat[1:(length(ytrain) - 365), "level"], col = "red")
lines(smo1$alphahat[1:(length(ytrain) - 365), "level"] +
      smo1_seas[1:(length(ytrain) - 365)], col = "blue")
lines(smo1$alphahat[1:(length(ytrain) - 365), "level"] +
      smo1$alphahat[1:(length(ytrain) - 365), "sea_dummy1"] +
      smo1_seas[1:(length(ytrain) - 365)], col = "green")

```



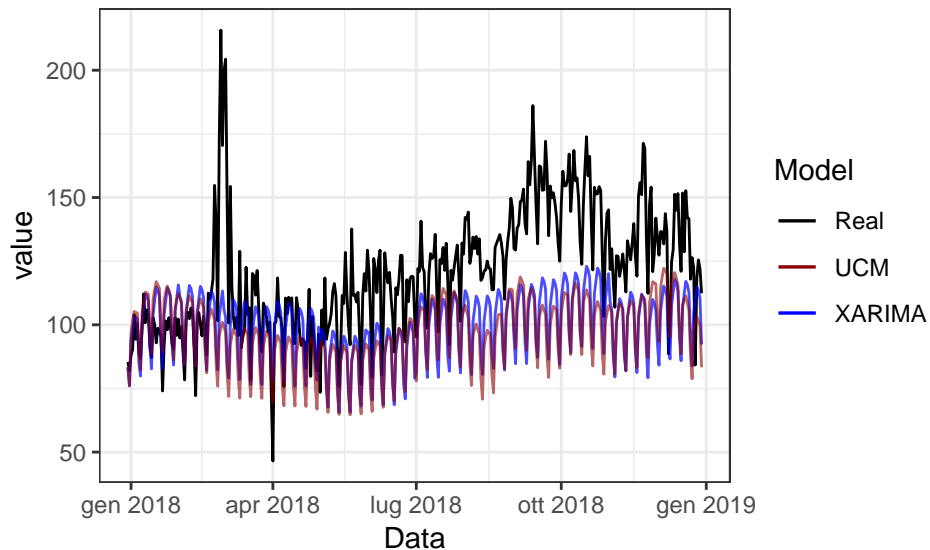
```

df[(((length(train)+1):length(ytrain)),) %>%
  mutate(UCM = smo1$muhat[(length(train)+1):length(ytrain)],
         XARIMA = as.numeric(pred_reg$mean),
         ARIMA = as.numeric(pred$mean)) %>%
  ggplot(aes(x=Data)) +
  geom_line(aes(y=value ,col="Real")) +
  #geom_line(aes(y=ARIMA, col="ARIMA"), linetype="F1") +

```

```
geom_line(aes(y=XARIMA, col="XARIMA"), alpha=0.7) +
geom_line(aes(y=UCM, col="UCM"), alpha=0.6) +
scale_color_manual(name = "Model",
                    values= c("Real"="black",
                              "UCM"="darkred",
                              #"ARIMA"="blue",
                              "XARIMA"="blue")) +

theme_bw()
```



```
mse_ucm <- sqrt(mean((test - smo1$muhat[(length(train)+1):length(ytrain)])^2))

print(paste0("MSE relativo modello ARIMA : ", mse_2))

## [1] "MSE relativo modello ARIMA : 27.2573932289143"

print(paste0("MSE relativo modello XARIMA : ", mse_reg))

## [1] "MSE relativo modello XARIMA : 28.7326528499392"

print(paste0("MSE relativo modello UCM : ", mse_ucm))

## [1] "MSE relativo modello UCM : 31.0755547902199"
```