# Pranav Kasela 846965

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

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

```
##
## 1 function (x, df1, df2, ncp, log = FALSE)
## 2 {
## 3     if (missing(ncp))
## 4         .Call(C_df, x, df1, df2, log)
## 5     else .Call(C_dnf, x, df1, df2, ncp, log)
## 6 }
```

```
in_df[in_df$Data=="2012-02-29" | in_df$Data=="2016-02-29",]
```

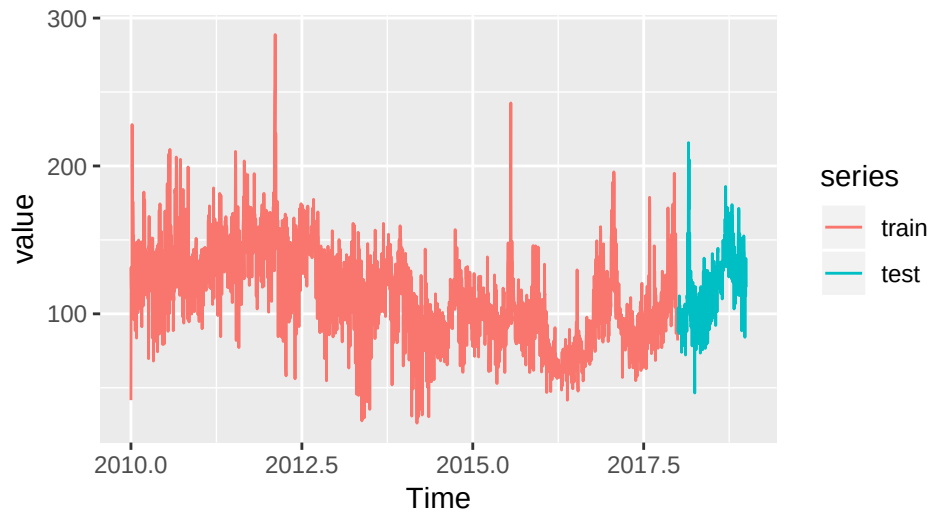
```
##            Data     value
## 790  2012-02-29 157.31243
## 2251 2016-02-29  73.72674
```

```
#remove the leap year dates
df <- in_df[-which(in_df$Data=="2012-02-29" | in_df$Data=="2016-02-29"),]
```

## Arima Models

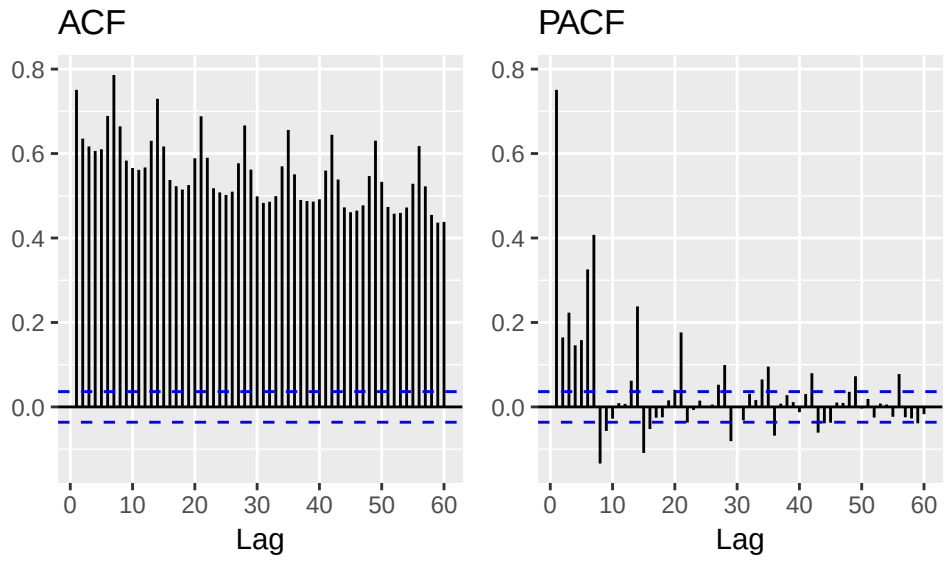
Splittiamo il dataset in train e test per verificare come va il modello su nuovi dati.

```
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))

autoplot(cbind(train, test)) + ylab("value")
```



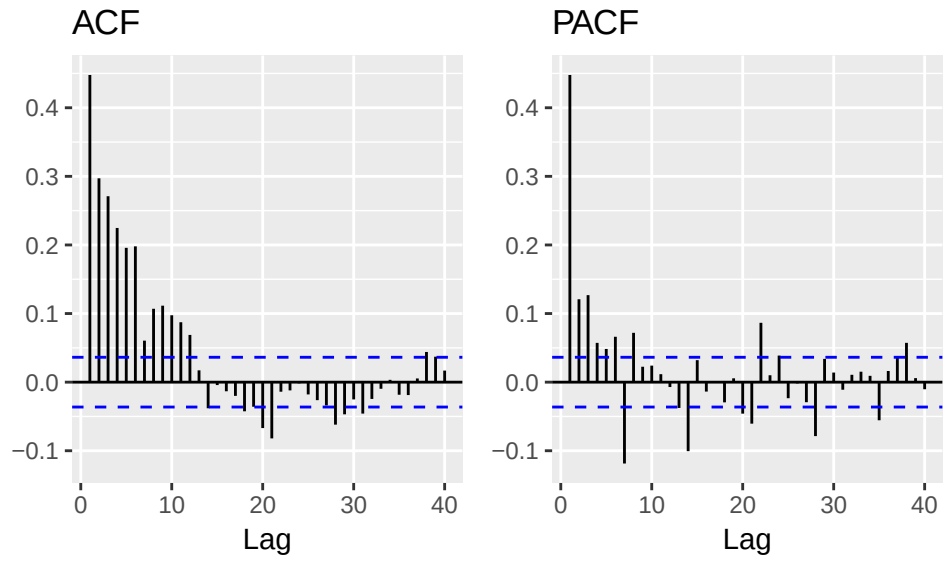

```
#This is the function ggtsdisplay of forecast package,
#but it has been modified so is doesn't plot the series,
#just the ACF and PACF plot, with the horizontal parameter
#the plot can be either horizontal or vertical
#The function has been simplified a lot, since we don't need
#all the complexity the original one has.
ggtsdisplay_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 ritrado stagionale che scende esponenzialemente al settimo ritardo, indicando la presenza di un $\text{SMA}(1)_7$ e vedendo ACF dai primi 2 ritardi stagionali ci convinciamo dell'esistenza di $\text{SMA}(1)_7$, inoltre vi è presente anche un $\text{SAR}(1)_7$. 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
```

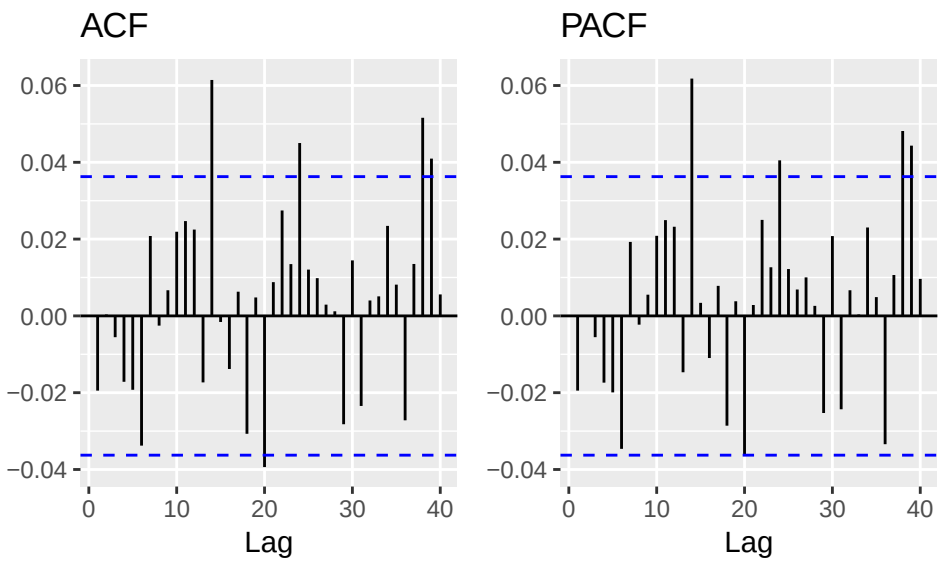
```
## Series: train
## ARIMA(0,0,0)(1,0,1)[7] with non-zero mean
## Box Cox transformation: lambda= 0.8062425
##
## Coefficients:
##         sar1     sma1     mean
##       0.9585  -0.5495  55.6316
## s.e.  0.0064   0.0228   1.3456
##
## sigma^2 estimated as 49.58:  log likelihood=-9845.76
## AIC=19699.52   AICc=19699.54   BIC=19723.44
```

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= 0.8062425
##
## Coefficients:
##          ar1     ar2     ar3     ar4     ar5     ar6    sar1     sma1
##       0.3985  0.0615  0.1136  0.0578  0.0374  0.1257  0.0662  -0.8963
## s.e.  0.0186  0.0200  0.0200  0.0201  0.0200  0.0192  0.0234   0.0127
##
## sigma^2 estimated as 35.41:  log likelihood=-9329.48
## AIC=18676.95   AICc=18677.01   BIC=18730.74
```

```
ggtsdisplay_2(mod2$residuals, lag.max = 40)
```




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

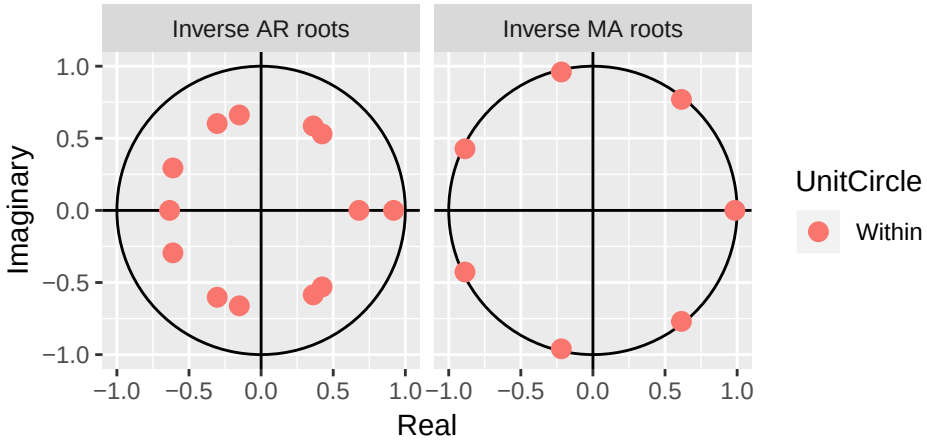
Usiamo il test Augmented Dickey-Fuller, che cerca radici unitarie nella serie, per verificare se la serie è stazionaria, con k che indica il ritardo autoregressivo, $H_0$ è la presenza di radici unitarie nella serie, $H_1$ è che la serie è stazionaria.

```
#Trying Augmeted 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)
```

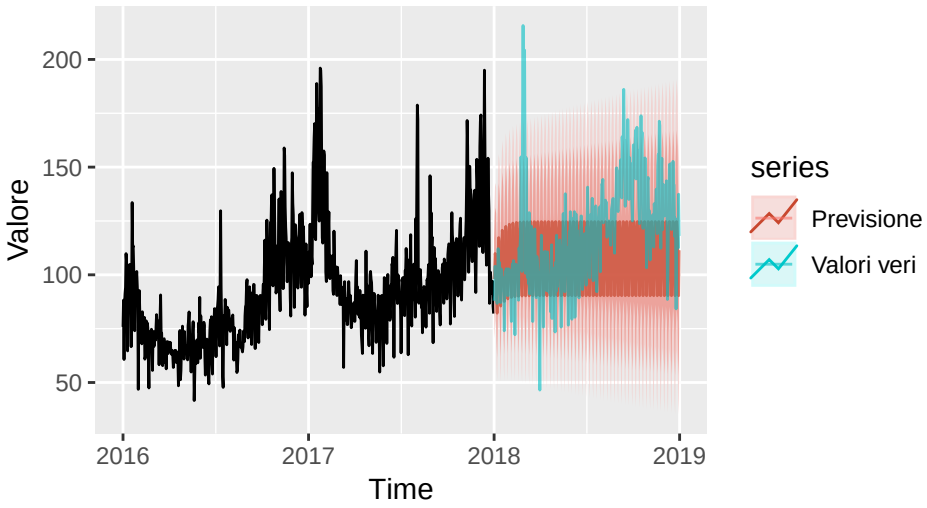
```
##
##  Augmented Dickey-Fuller Test
##
## data:  mod2$residuals
## Dickey-Fuller = -19.907, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

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)
```



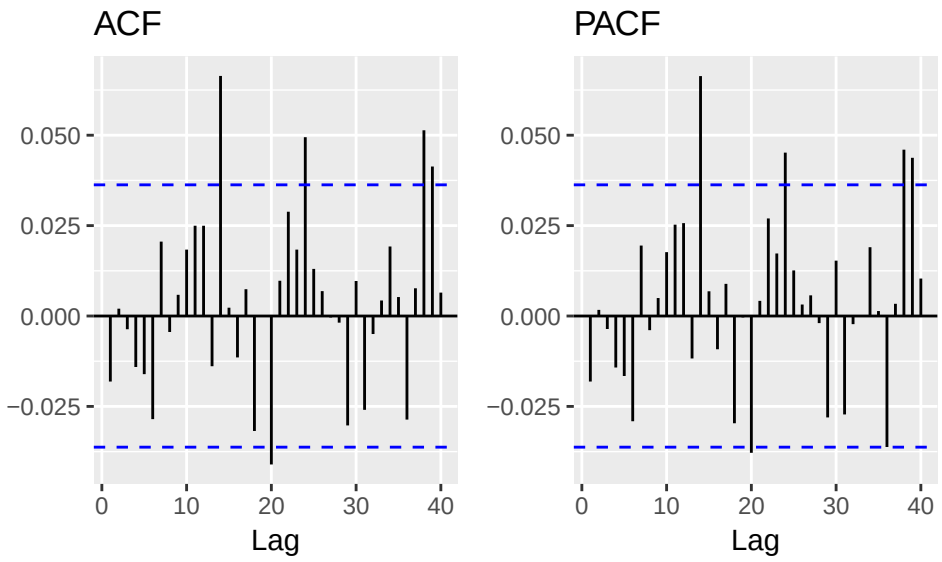

```
pred <- forecast(mod2, h=365)

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")
```



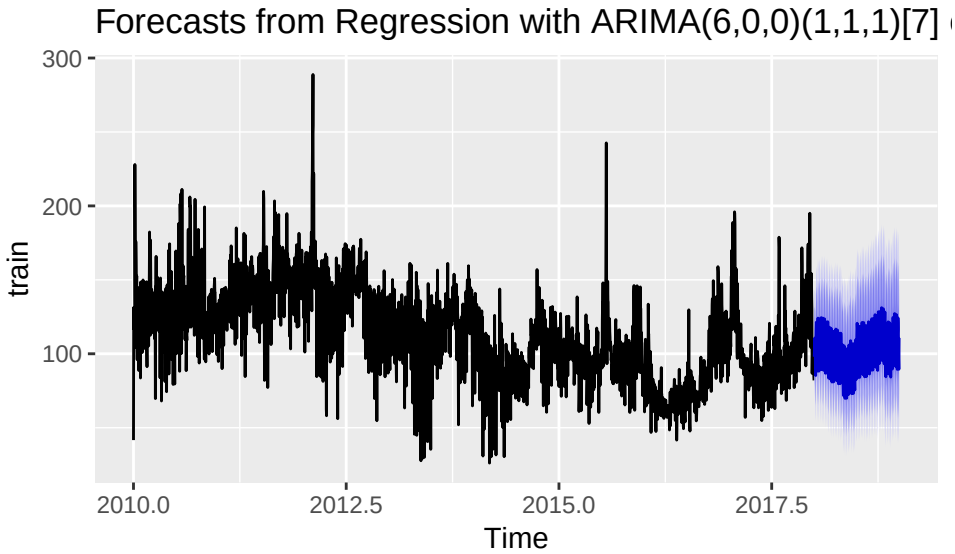

Mettiamo regressori dummy mensili e ogni 4 settimane.

```
#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
```

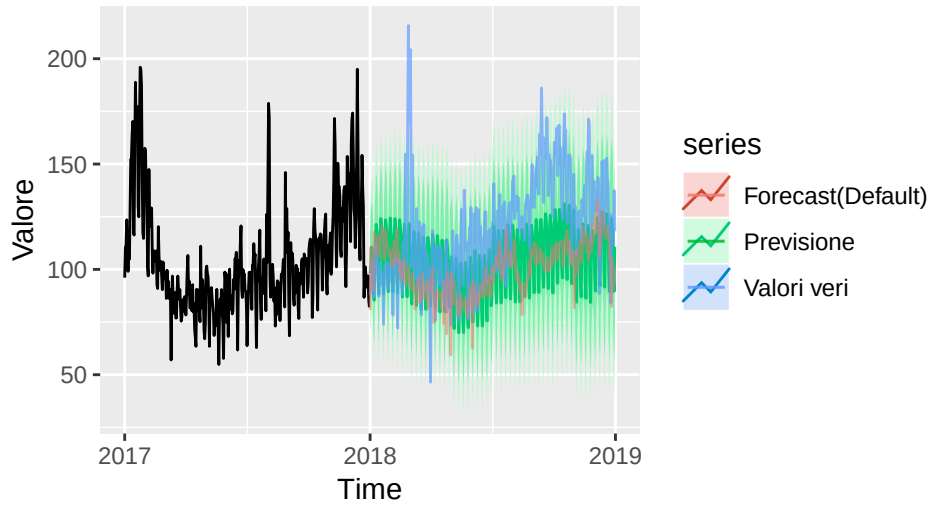
```r
#eliminiamo la prima e l'ultima colonna, per evitare multicollinearità
xreg <- more_reg[,2:(ncol(more_reg)-1)]

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)
```




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




```r
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.

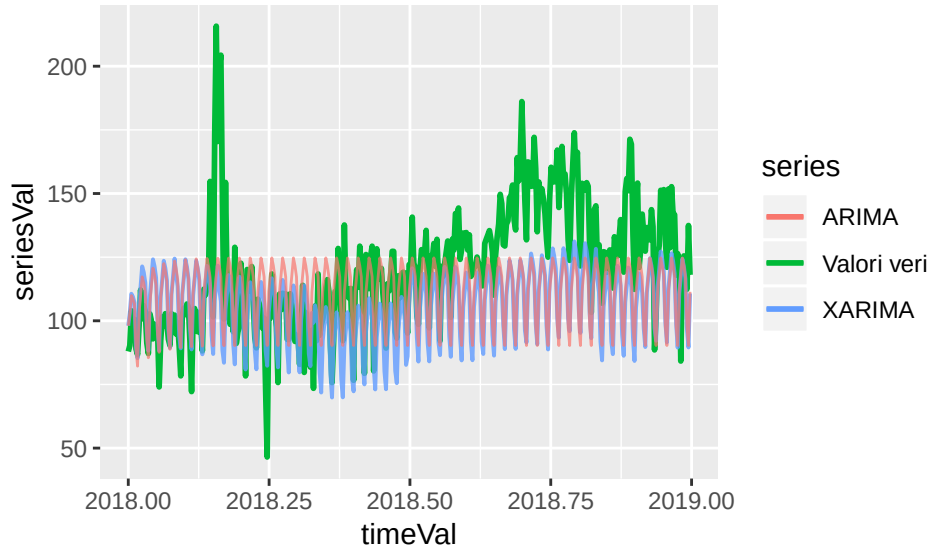
```
mod_reg_y <- forecast(mod1_reg, h=365,
                 xreg=xreg[(nrow(df)-365+1):(length(train)+365),])$mean
mod2_y <- forecast(mod2, h=365)$mean

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 : 24.4995769284025"
```

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

```
## [1] "MSE relativo modello con regressori : 25.5021351974869"
```

```
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

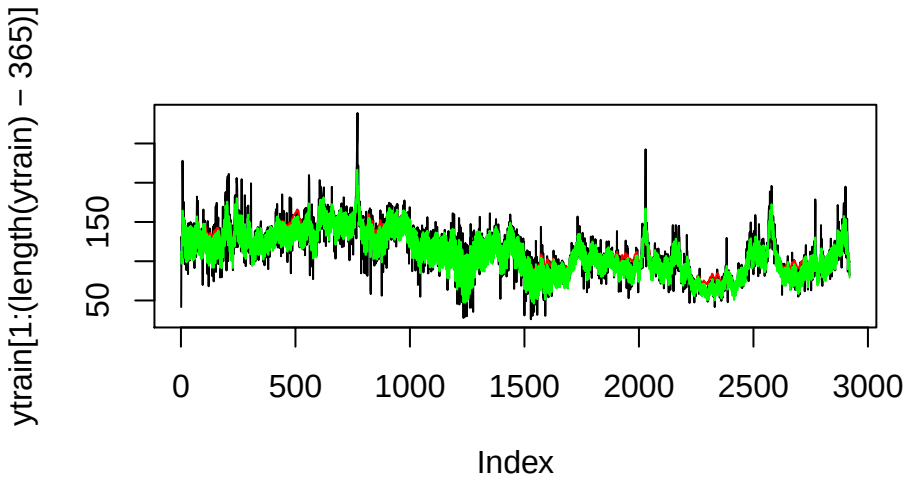
```
library(KFAS)
```

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

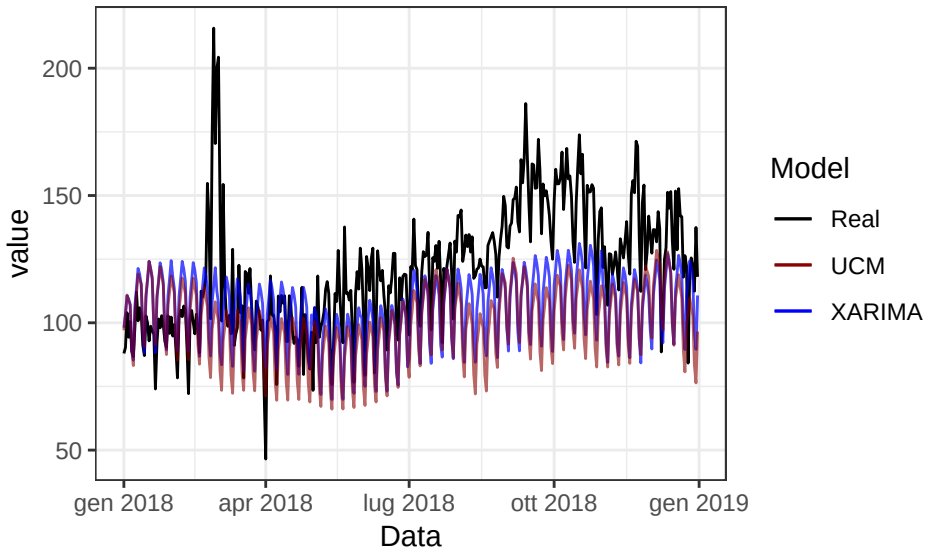
```
mod1 <- SSModel(ytrain ~ 0 +
                  SSMtrend(1, NA) +
                  SSMseasonal(7, NA, "dummy") +
                  SSMseasonal(365, 0, "trig",
                              harmonics = 1:12),
                H = NA)

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 : 24.4995769284025"
```

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

```
## [1] "MSE relativo modello XARIMA : 25.5021351974869"
```

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

```
## [1] "MSE relativo modello UCM : 29.3430919663313"
```