paper_final

```r
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_current$get(c(
  "cache",
  "cache.path",
  "cache.rebuild",
  "dependson",
  "autodep"
))
```

```
## $cache
## [1] 0
##
## $cache.path
## [1] "RMD_final_cache/latex/"
##
## $cache.rebuild
## [1] FALSE
##
## $dependson
## NULL
##
## $autodep
## [1] FALSE
```

```r
knitr::opts_knit$set(root.dir = '/Users/valerioschips/Desktop/DSLab/Progetto/Dati\ Energia')
```

DSLab Report

Configurazione

Preferences -> R Markdown -> Evaluate Chunks in Directory: Current

Caricamento Librerie

```r
library(readxl)
library(dplyr)
library(ggplot2)
library(forecast)
library(tseries)
```

Caricamento Dati

```r
setwd('/Users/valerioschips/Desktop/DSLab/Progetto/Dati\ Energia')
```

Variabili globali contenti i nomi dei file e delle colonne

```r
g_build <- c("U1", "U6")
g_year <- c("18", "19", "20")
g_month <- c("01","02","03","04","05","06","07","08","09","10","11","12")
g_columns_subset <- c("POD", "DATA", "ORA", "FL_ORA_LEGALE", "CONSUMO_ATTIVA_PRELEVATA", "CONSUMO_REATTI
g_columns_num <- c("CONSUMO_ATTIVA_PRELEVATA", "CONSUMO_REATTIVA_INDUTTIVA_PRELEVATA", "POTENZA_MASSIMA
g_aggregate_level <- c("Def", "Hour", "Day", "Month", "Year")
g_meteo_column <- c("tavg", "tmin", "tmax", "prcp", "pres")
```

Funzioni per il caricamento dei Datase, Standardizzazione delle colonne e Aggregazione

```r
make_numeric_column <- function(p_df){
  for (x in g_columns_num){
    p_df[,x] <- as.numeric(unlist(p_df[,x]))
  }
  return(p_df)
}

check_columns_subset <- function(p_df){
  for (x in g_columns_subset){
    if (!(x %in% colnames(p_df))){
      p_df[, x] <- NA
    }
  }
  return(p_df)
}

single_energy_table <- function(build=NULL, year=NULL, month=NULL){
  path <- c(build, year)
  path <- paste(path, collapse = "/")
  tmp_path <- c(path, "/" ,month, ".xlsx")
  tmp_path <- paste(tmp_path, collapse = "")
  ret_df <- read_excel(tmp_path)
  ret_df <- check_columns_subset(ret_df)
  ret_df <- ret_df[,g_columns_subset]
  ret_df<-make_numeric_column(ret_df)
  ret_df$DATA <- as.Date(as.character(ret_df$DATA), "%Y%m%d")
  ret_df$WDAY <- as.POSIXlt(ret_df$DATA)$wday
  ret_df$ENERGIA_CONSUMATA <- ret_df$CONSUMO_ATTIVA_PRELEVATA * 900
  return(ret_df)
}

single_build_energy_table <- function(build=NULL, year=NULL, month=NULL){
  if (is.null(year)) {
    if (is.null(month)){
      month_list <- g_month
    }else{
      month_list <- month
    }
    ret_df <- single_energy_table(build, g_year[1], month_list[1])
    st_index_month <- 2
    for (y in 1:length(g_year)) {
      if (length(month_list) >= st_index_month){
        for (x in st_index_month:length(month_list)) {
          tmp_data <- single_energy_table(build, g_year[y], month_list[x])
```

```
          ret_df <- rbind(ret_df, tmp_data)
        }
      }
      st_index_month <- 1
    }
    return(ret_df)
  }
  if (is.null(month)) {
    if (is.null(year)){
      year_list <- g_year
    }else{
      year_list <- year
    }
    ret_df <- single_energy_table(build, year_list[1], g_month[1])
    st_index_month <- 2
    for (y in 1:length(year_list)) {
      for (x in st_index_month:length(g_month)) {
        tmp_data <- single_energy_table(build, year_list[y], g_month[x])
        ret_df <- rbind(ret_df, tmp_data)
      }
      st_index_month <- 1
    }
    return(ret_df)
  }

  ret_df <- single_energy_table(build, year, month[1])
  if(length(month) >1){
    for (x in 2:length(month)) {
      tmp_data <- single_energy_table(build, year, month[x])
      ret_df <- rbind(ret_df, tmp_data)
    }
  }
  return(ret_df)
}

both_build_energy_table <- function(year=NULL, month=NULL, aggregate = "Def"){
    U1 <- single_build_energy_table(g_build[1], year, month)
    U6 <- single_build_energy_table(g_build[2], year, month)
    if (aggregate != "Def"){
      U1 <- time_aggregate(U1, aggregate)
      U6 <- time_aggregate(U6, aggregate)
    }
    if (aggregate == "Month") {
      ret_df <- merge(U1, U6, by = c("DATA"), all = TRUE, suffixes= c("_U1", "_U6"))
    }else if(aggregate == "Day"){
      ret_df <- merge(U1, U6, by = c("DATA", "WDAY", "FL_ORA_LEGALE"), all = TRUE, suffixes= c("_U1", "
    }else{
      ret_df <- merge(U1, U6, by = c("DATA", "ORA", "WDAY", "FL_ORA_LEGALE"), all = TRUE, suffixes= c("
    }
    return(ret_df)
}
time_aggregate <- function(p_df, type){
  p_df$DAY <- format(p_df$DATA, "%d")
```

```r
  p_df$MONTH <- format(p_df$DATA, "%m")
  p_df$YEAR <- format(p_df$DATA, "%Y")
  if ("ORA" %in% colnames(p_df)){
    p_df$ORA <- as.integer(p_df$ORA/10000)
  }
  if (type == "Hour"){
    p_df_copy <- p_df %>% group_by(YEAR, MONTH, DAY, ORA) %>% summarise(DATA = min(DATA), CONSUMO_ATTIVA
    p_df_copy <- subset(p_df_copy, select = -c(DAY))
  }else if(type == "Day"){
    p_df_copy <- p_df %>% group_by(YEAR, MONTH, DAY) %>% summarise(DATA = min(DATA), CONSUMO_ATTIVA_PREL
    p_df_copy <- subset(p_df_copy, select = -c(DAY))
  }else if(type == "Month"){
    p_df_copy <- p_df %>% group_by(YEAR, MONTH) %>% summarise(DATA = min(DATA), CONSUMO_ATTIVA_PRELEVATA
  }else if(type == "dayMonth"){
    p_df_copy <- p_df %>% group_by(YEAR, MONTH) %>% summarise(DATA = min(DATA), CONSUMO_ATTIVA_PRELEVATA
  }
  p_df_copy$DATA <- as.Date(p_df_copy$DATA, "%Y%m%d")
  p_df_copy <- subset(p_df_copy, select = -c(YEAR, MONTH))

  return(p_df_copy)
}

weather_data_load<-function(p_df, year){
  path <- c("METEO", year[1])
  path <- paste(path, collapse = "/")
  path <- c(path, "xlsx")
  path <- paste(path, collapse = ".")
  ret_df <- read_excel(path)
  if (length(year) > 1){
    for (x in 2:length(year)){
      path <- c("METEO", year[x])
      path <- paste(path, collapse = "/")
      path <- c(path, "xlsx")
      path <- paste(path, collapse = ".")
      tmp_data <- read_excel(path)
      ret_df <- rbind(ret_df, tmp_data)
    }
  }
  for (x in g_meteo_column){
    ret_df[,x] <- as.numeric(unlist(ret_df[,x]))
  }
  ret_df$DATA <- as.Date(as.character(as.POSIXct(ret_df$DATA, 'GMT')))
  ret_df <- merge(p_df, ret_df, by = c("DATA"), all.x = TRUE)
  return(ret_df)
}
```

Funzione per aggiungere la variabile aperto chiuso ad un dataframe.

```r
add_dummy_open <- function(p_df){
p_df$WDAY <- as.POSIXlt(p_df$DATA)$wday
p_df$aperto <- ifelse(p_df$WDAY == 0, 0, 1)
p_df[(p_df$DATA >= "2019-01-01") & (p_df$DATA <= "2019-01-04"),]$aperto <- 0
p_df[p_df$DATA == "2019-01-06",]$aperto <- 0
p_df[(p_df$DATA >= "2019-03-07") & (p_df$DATA <= "2019-03-08"),]$aperto <- 0
```

```
p_df[(p_df$DATA >= "2019-04-18") & (p_df$DATA <= "2019-04-19"),]$aperto <- 0
p_df[(p_df$DATA >= "2019-04-21") & (p_df$DATA <= "2019-04-23"),]$aperto <- 0
p_df[p_df$DATA == "2019-04-25",]$aperto <- 0
p_df[p_df$DATA == "2019-05-01",]$aperto <- 0
p_df[p_df$DATA == "2019-06-02",]$aperto <- 0
p_df[(p_df$DATA >= "2019-08-11") & (p_df$DATA <= "2019-08-16"),]$aperto <- 0
p_df[p_df$DATA == "2019-11-01",]$aperto <- 0
p_df[(p_df$DATA > "2019-12-07") & (p_df$DATA <= "2019-12-08"),]$aperto <- 0
p_df[(p_df$DATA >= "2019-12-21") & (p_df$DATA <= "2019-12-31"),]$aperto <- 0

p_df[(p_df$DATA >= "2018-01-01") & (p_df$DATA <= "2018-01-07"),]$aperto <- 0
p_df[(p_df$DATA >= "2018-02-15") & (p_df$DATA <= "2018-02-18"),]$aperto <- 0
p_df[(p_df$DATA >= "2018-03-29") & (p_df$DATA <= "2018-04-03"),]$aperto <- 0
p_df[p_df$DATA == "2018-04-25",]$aperto <- 0
p_df[p_df$DATA == "2018-05-01",]$aperto <- 0
p_df[p_df$DATA == "2018-06-02",]$aperto <- 0
p_df[p_df$DATA == "2018-08-15",]$aperto <- 0
p_df[p_df$DATA == "2018-11-01",]$aperto <- 0
p_df[(p_df$DATA >= "2018-12-07") & (p_df$DATA <= "2018-12-08"),]$aperto <- 0
p_df[(p_df$DATA > "2018-12-23") & (p_df$DATA <= "2018-12-30"),]$aperto <- 0


p_df[(p_df$DATA >= "2020-01-01") & (p_df$DATA <= "2020-01-07"),]$aperto <- 0
p_df[(p_df$DATA >= "2020-02-27") & (p_df$DATA <= "2020-02-29"),]$aperto <- 0
p_df[(p_df$DATA >= "2020-04-09") & (p_df$DATA <= "2020-04-14"),]$aperto <- 0
p_df[p_df$DATA == "2020-04-25",]$aperto <- 0
p_df[p_df$DATA == "2020-05-01",]$aperto <- 0
p_df[p_df$DATA == "2020-06-02",]$aperto <- 0
p_df[p_df$DATA == "2020-08-15",]$aperto <- 0
p_df[p_df$DATA == "2020-11-01",]$aperto <- 0
p_df[(p_df$DATA >= "2020-12-23") & (p_df$DATA <= "2020-12-31"),]$aperto <- 0
return(p_df)
}
```

Cap 1.5 Missing Value

Previsione U6 Giugno

Funzione Sliding Window

```
sliding_window <- function(p_df, p_column, p_window, p_st_date, p_foreward, p_lenght){

  p_lenght <- p_lenght + 14

  b <- .5
  p_weight <- c(.5)
  for (a in 1:(p_foreward - 2)){
    b <- b/2
    p_weight <- append(p_weight, b)
    a <- a+1
  }
  p_weight <- append(p_weight, 1-sum(p_weight))

  p_df <- p_df[1:(p_window+p_foreward-1), c("DATA", p_column)]
```

```r
  p_st_date <- as.Date(p_st_date,format="%Y-%m-%d")
  end_date <- as.Date(p_st_date,format="%Y-%m-%d") + p_window - 1


  ts_st_date <- c(as.numeric(format(as.Date(p_st_date,format="%Y-%m-%d"), format = "%m")))
  ts_st_date <- append(ts_st_date, as.numeric(format(as.Date(p_st_date,format="%Y-%m-%d"), format = "%d
  ts_end_date <- c(as.numeric(format(as.Date(end_date,format="%Y-%m-%d"), format = "%m")))
  ts_end_date <- append(ts_end_date, as.numeric(format(as.Date(end_date,format="%Y-%m-%d"), format = "%

  result_df <- p_df
  tmp_df <- data.frame(DATA= seq((end_date+p_foreward), (end_date +(p_lenght - p_foreward)), "day"))
  tmp_df[,p_column] <- 0
  result_df <- rbind(result_df, tmp_df)

  df_row <- nrow(result_df)

  start_df <- result_df[(result_df$DATA >= p_st_date) & (result_df$DATA <= end_date),]
  for (i in (p_window+1):(df_row-1)){
    analysis_ts <- ts(start_df[, p_column], frequency = 7, start = c(1,1))
    tbats_fitted <- auto.arima(analysis_ts)
    fore <- forecast(tbats_fitted, h=p_foreward)


    tmp_df <- result_df[(i):(i+p_foreward-1), c("DATA", p_column)]
    weighted_val <- p_weight*as.vector(fore$mean)
    tmp_df$fitted_v <- weighted_val
    names(tmp_df)[length(names(tmp_df))]<-paste0("V", i)
    result_df <- merge(result_df, tmp_df, all.x = TRUE)

    if( (i > (p_window + p_foreward-1))  ){
      result_df[i, p_column] <- rowSums(result_df[i,-c(1,2)], na.rm = TRUE)
    }

    end_date <- end_date + 1
    p_st_date <- p_st_date + 1
    ts_st_date <- c(as.numeric(format(as.Date(p_st_date,format="%Y-%m-%d"), format = "%m")))
    ts_st_date <- append(ts_st_date, as.numeric(format(as.Date(p_st_date,format="%Y-%m-%d"), format = "
    ts_end_date <- c(as.numeric(format(as.Date(end_date,format="%Y-%m-%d"), format = "%m")))
    ts_end_date <- append(ts_end_date, as.numeric(format(as.Date(end_date,format="%Y-%m-%d"), format =

    start_df <- result_df[(result_df$DATA >= p_st_date) & (result_df$DATA <= end_date),]
  }
  return(result_df)
}
```

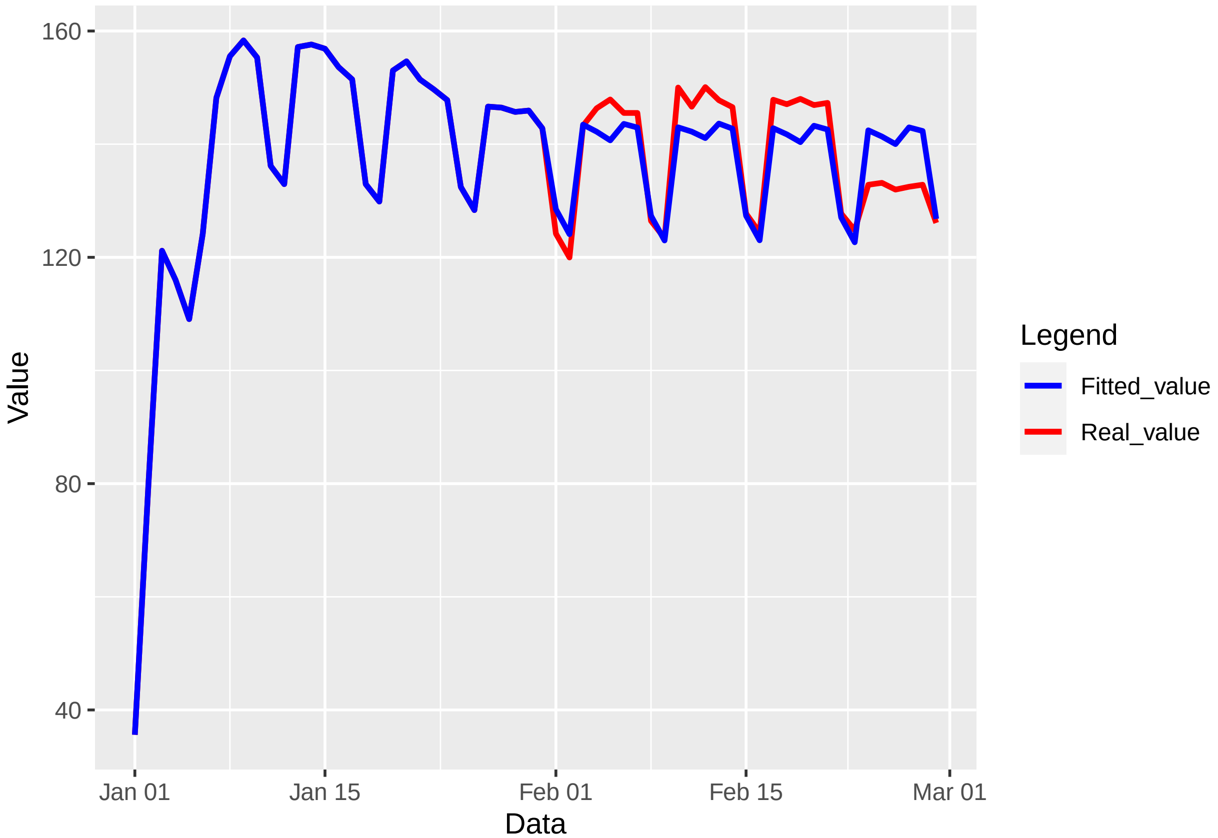
```r
u6_df <- single_build_energy_table("u6", year = c("18","19","20"))
u6_df <- time_aggregate(u6_df, "Day")
u6_df$Target_Column <- u6_df$CONSUMO_ATTIVA_PRELEVATA_AVG
tag_name <- "CONSUMO_ATTIVA_PRELEVATA_AVG"
```

Valutiamo la previsione su Febbraio:

```r
february_predict_sw <- sliding_window(u6_df, "Target_Column", 755, "2018-01-01", 7, 29)
```

```
february_predict_sw <- data.frame(Data=february_predict_sw[,1], Real_value=u6_df[1:nrow(february_predic
february_predict_sw$acc <- with(february_predict_sw, abs(Real_value - Fitted_value)/Real_value * 100)
```

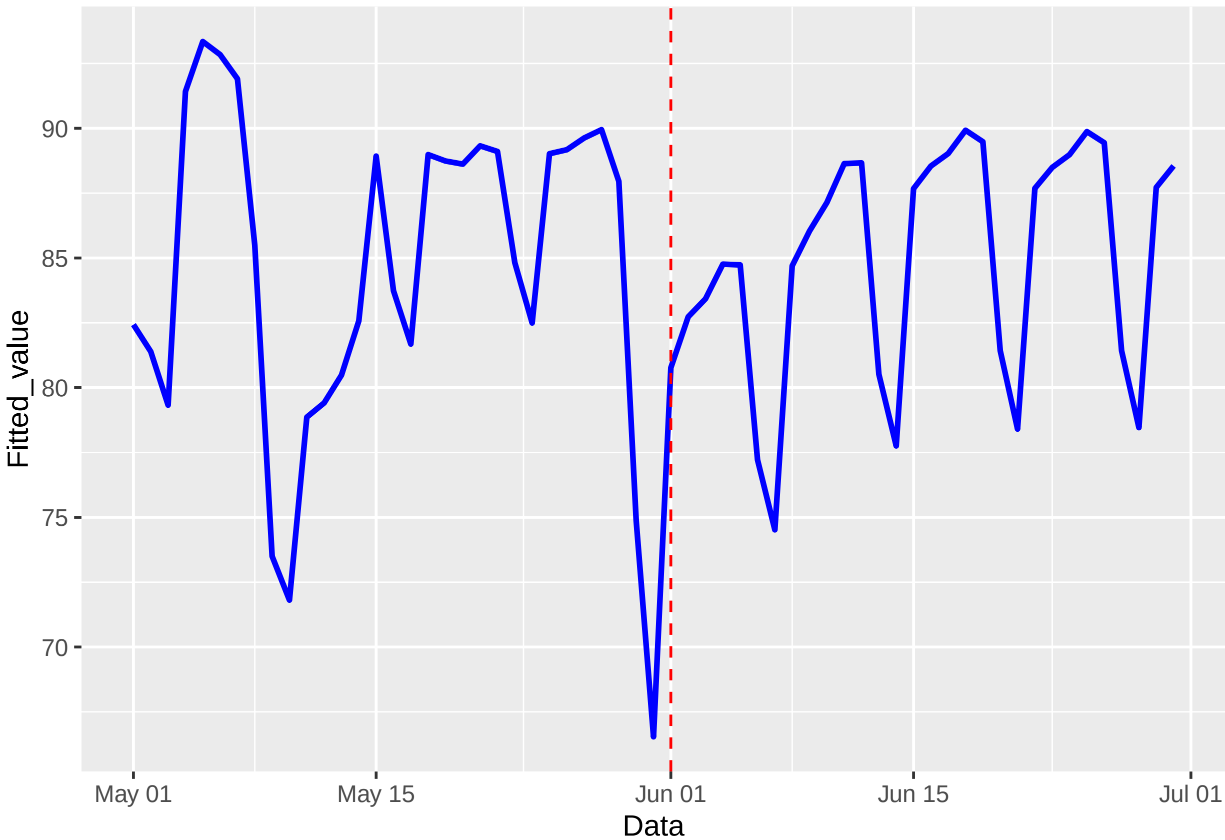
```
february_predict_sw <- february_predict_sw[february_predict_sw$Data >= "2020-01-01" & february_predict_s
legend <- c("Real_value" = "red", "Fitted_value" = "blue")
ggplot()+
  geom_line(data = february_predict_sw, aes(x= Data, y=Real_value, group=1, color="Real_value"), size=1)
  geom_line(data = february_predict_sw, aes(x= Data, y=Fitted_value, group=1, color="Fitted_value"), si
  labs(x = "Data", y = "Value", color = "Legend") +
  scale_colour_manual(values=legend)
```



Prevision Giugno:

```
june_predict_sw <- sliding_window(u6_df, "Target_Column", 876, "2018-01-01", 7, 30)
june_predict_sw <- data.frame(Data=june_predict_sw[,1], Real_value=u6_df[1:nrow(june_predict_sw),]$Targ
june_predict_sw$acc <- with(june_predict_sw, abs(Real_value - Fitted_value)/Real_value * 100)
```

```
june_predict_sw <- june_predict_sw[june_predict_sw$Data >= "2020-05-01" & june_predict_sw$Data < "2020-
ggplot()+
  geom_line(data = june_predict_sw, aes(x= Data, y=Fitted_value, group=1), color="blue", size=1)+
  geom_vline(xintercept = as.Date("2020-06-01"), color = "red", linetype="dashed")
```



Sostituiamo i dati mancanti nel mese Giugno 2020 - U6:

```
june_predict_subset_sw <- june_predict_sw[june_predict_sw$Data >= "2020-06-01" & june_predict_sw$Data <

colnames(june_predict_subset_sw) <- c("DATA",tag_name)

u6_df <- u6_df[!(u6_df$DATA >= "2020-06-01" & u6_df$DATA < "2020-07-01"),]

for (x in colnames(u6_df)){
  if (!(x %in% colnames(june_predict_subset_sw))){
    june_predict_subset_sw[, x] <- NA
  }
}
u6_df <- rbind(u6_df, june_predict_subset_sw)
u6_df[(u6_df$DATA >= "2020-06-01" & u6_df$DATA < "2020-07-01"),]$ENERGIA_CONSUMATA <- u6_df[(u6_df$DATA
u6_df<-u6_df[order(u6_df$DATA),]
```

```
rm(june_predict_subset_sw)
```

Cap 2.1 Analisi grafiche e Test

```
u1 <- single_build_energy_table(build = "U1", year = c("18", "19", "20"))
u1 <- time_aggregate(u1, "Day")
u6 <- u6_df
both_build_df <- merge(u1, u6, by = c("DATA"), all = TRUE, suffixes= c("_U1", "_U6"))
tag_name <- "CONSUMO_ATTIVA_PRELEVATA_AVG"
```

```r
both_build_ts <- ts(both_build_df, frequency =365, start = c(2018,1), end = c(2021,1))

autoplot(both_build_ts[,c(paste0(tag_name, "_U1"),paste0(tag_name, "_U6"))])+
         labs(x="Years", y="Kilo-watt", title="Confronto U1-U6")+  theme(legend.position="none")
```

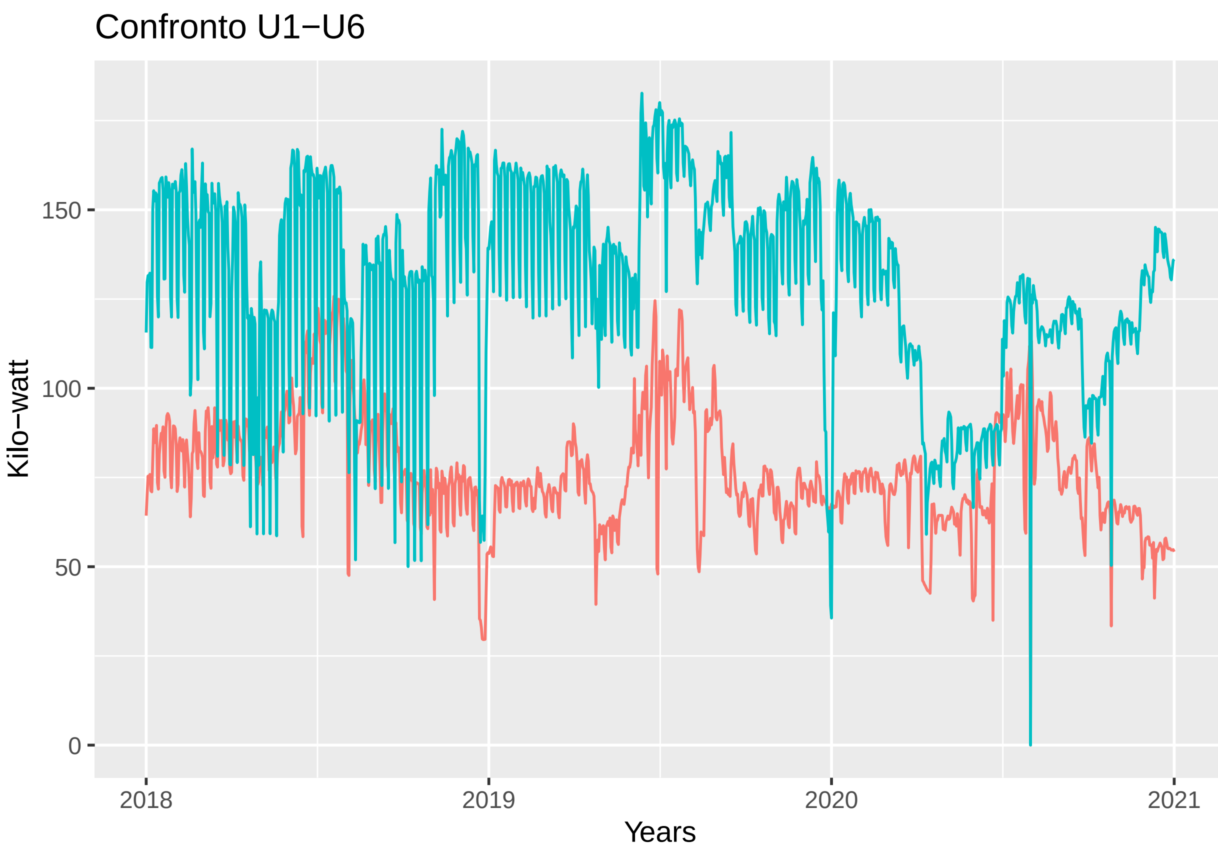


Figura 1: Confronto U1-U6

```r
u1 <- time_aggregate(u1, "dayMonth")
u6 <- time_aggregate(u6, "dayMonth")
ts_month_u1 <- ts(u1[, tag_name], frequency = 12, start = c(2018,1), end = c(2020,12))
ts_month_u6 <- ts(u6[, tag_name], frequency = 12, start = c(2018,1), end = c(2020,12))
```

```r
ggseasonplot(ts_month_u1, year.labels=TRUE, year.labels.left=TRUE)
```

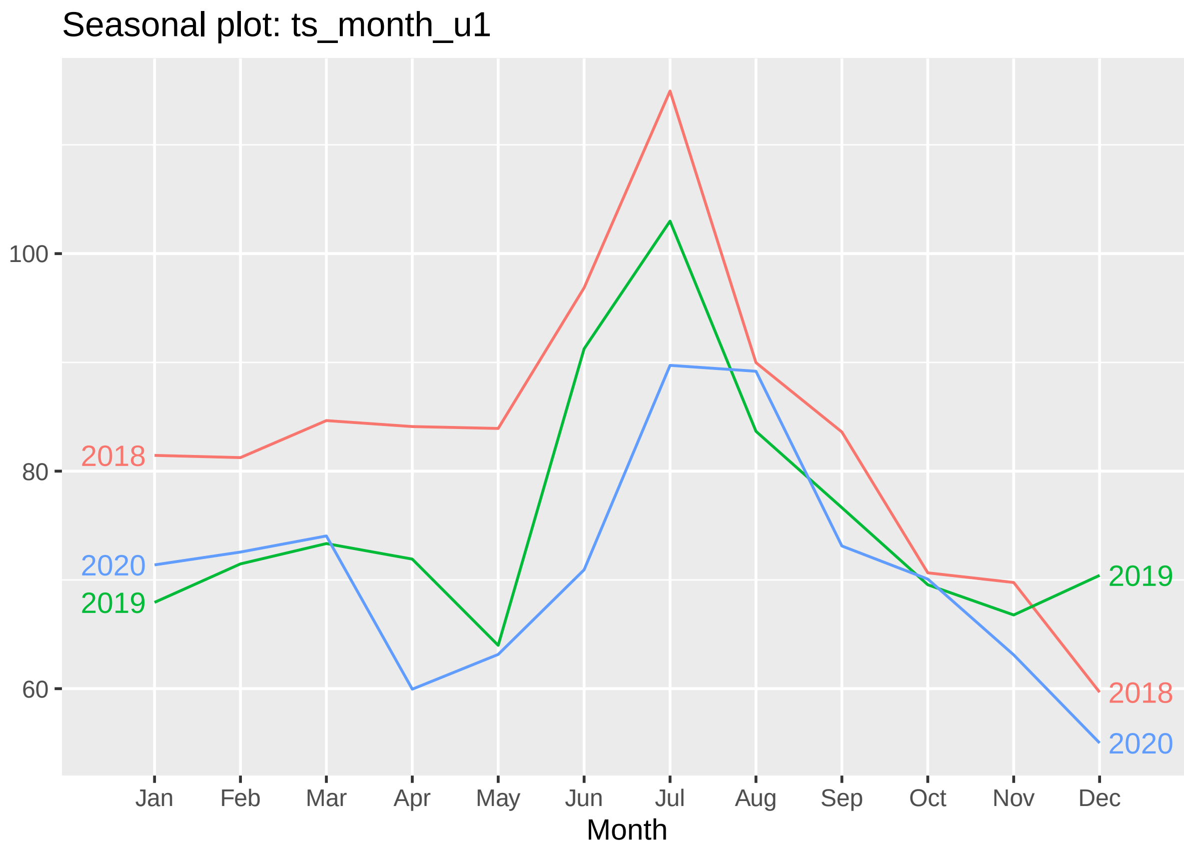


Figura 2: Seasonal plot U1

```
ggseasonplot(ts_month_u6, year.labels=TRUE, year.labels.left=TRUE)
```

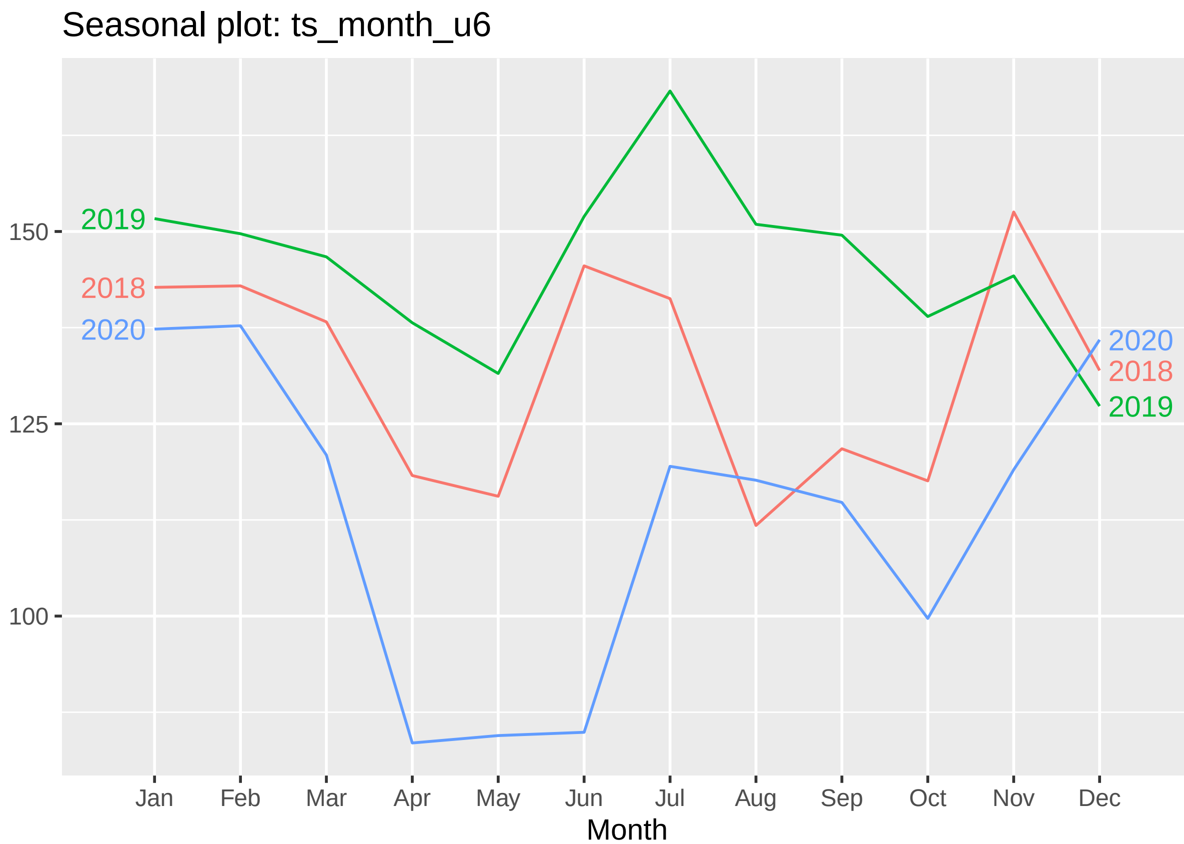
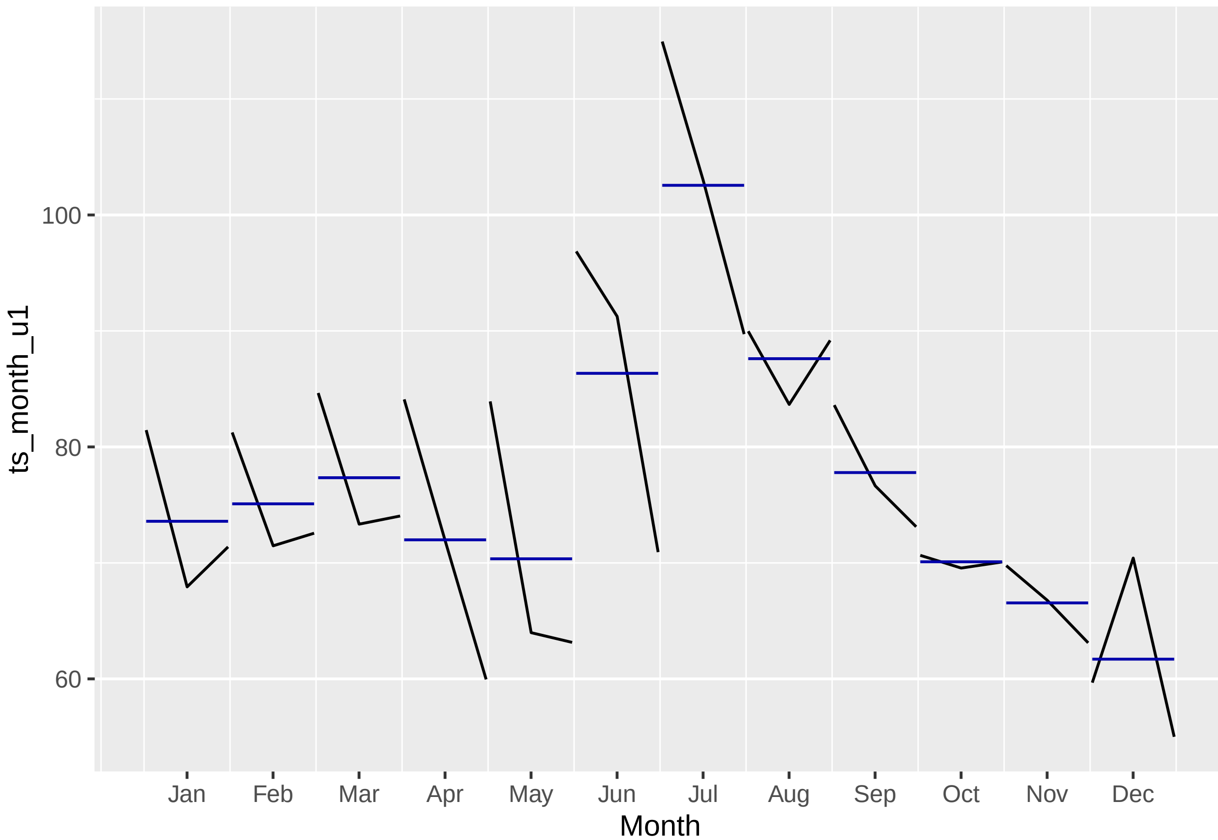


Figura 3: Seasonal plot U6

```
ggsubseriesplot(ts_month_u1)
```




```
ggsubseriesplot(ts_month_u6)
```

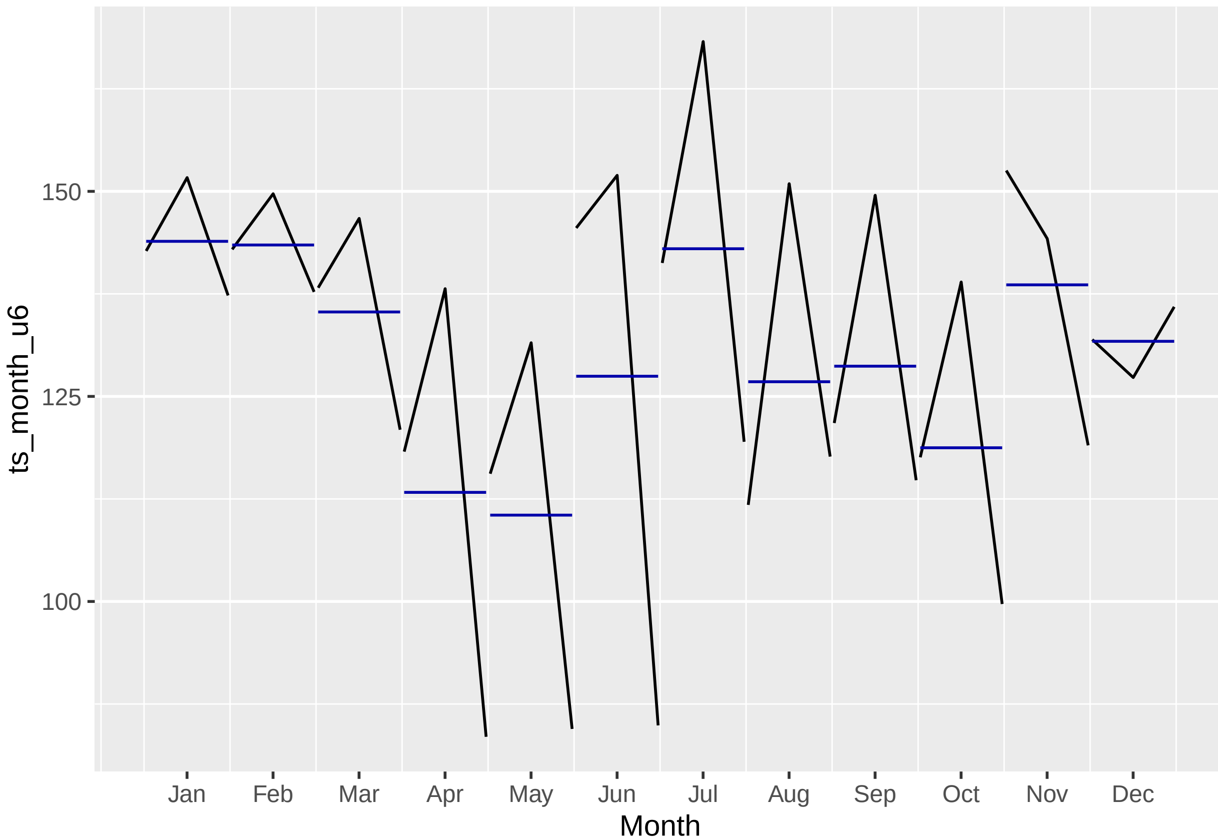
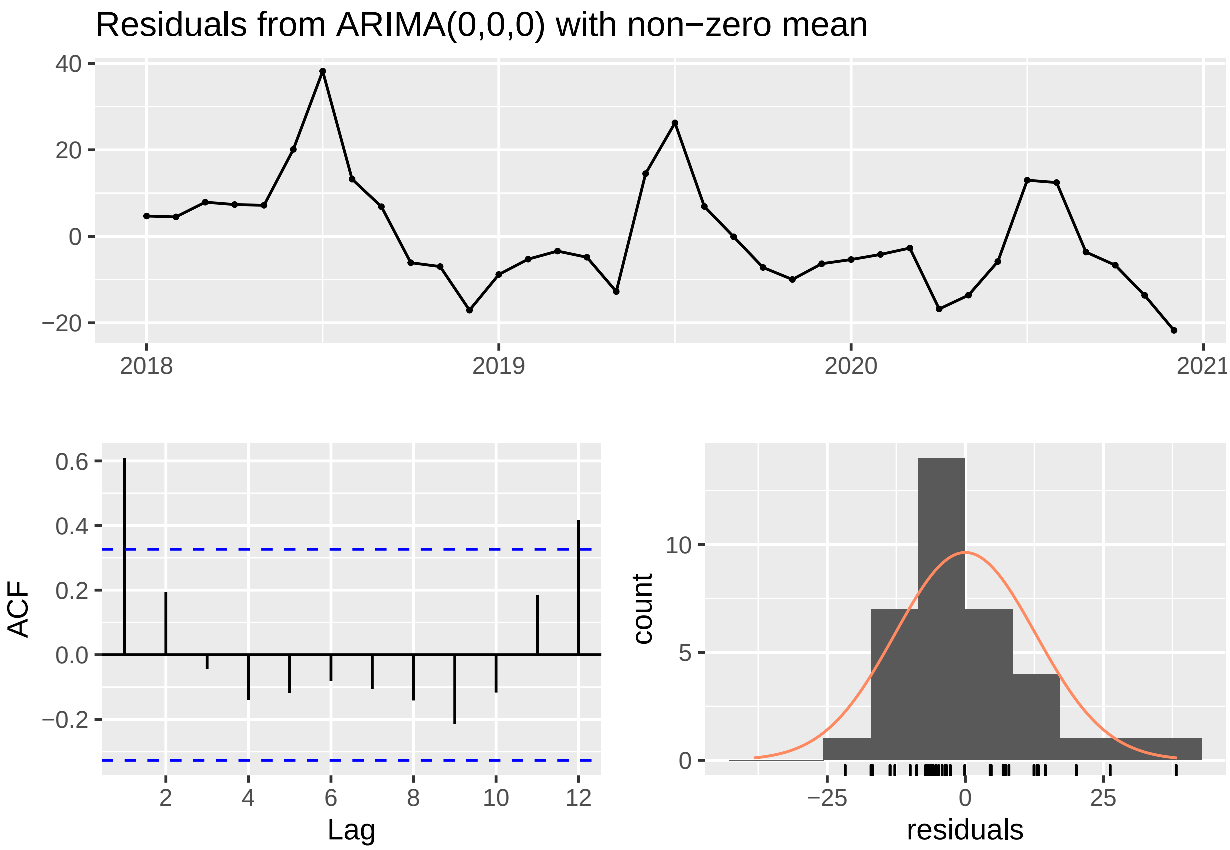


Figura 4: Subseries plot U1-U6

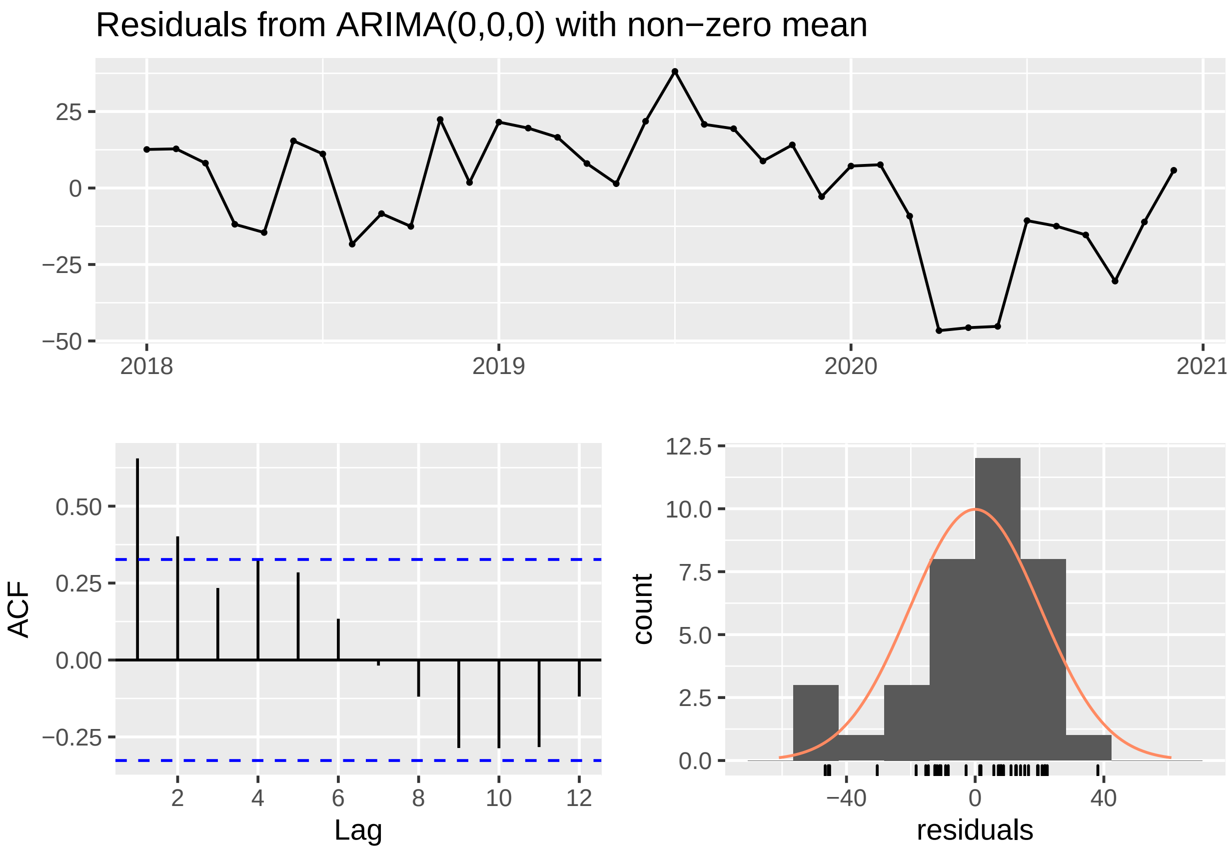
```
checkresiduals(arima(ts_month_u1))
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0) with non-zero mean
## Q* = 18.36, df = 6, p-value = 0.005393
##
## Model df: 1.   Total lags used: 7
```

Figura 5: Residuals Analysis U1

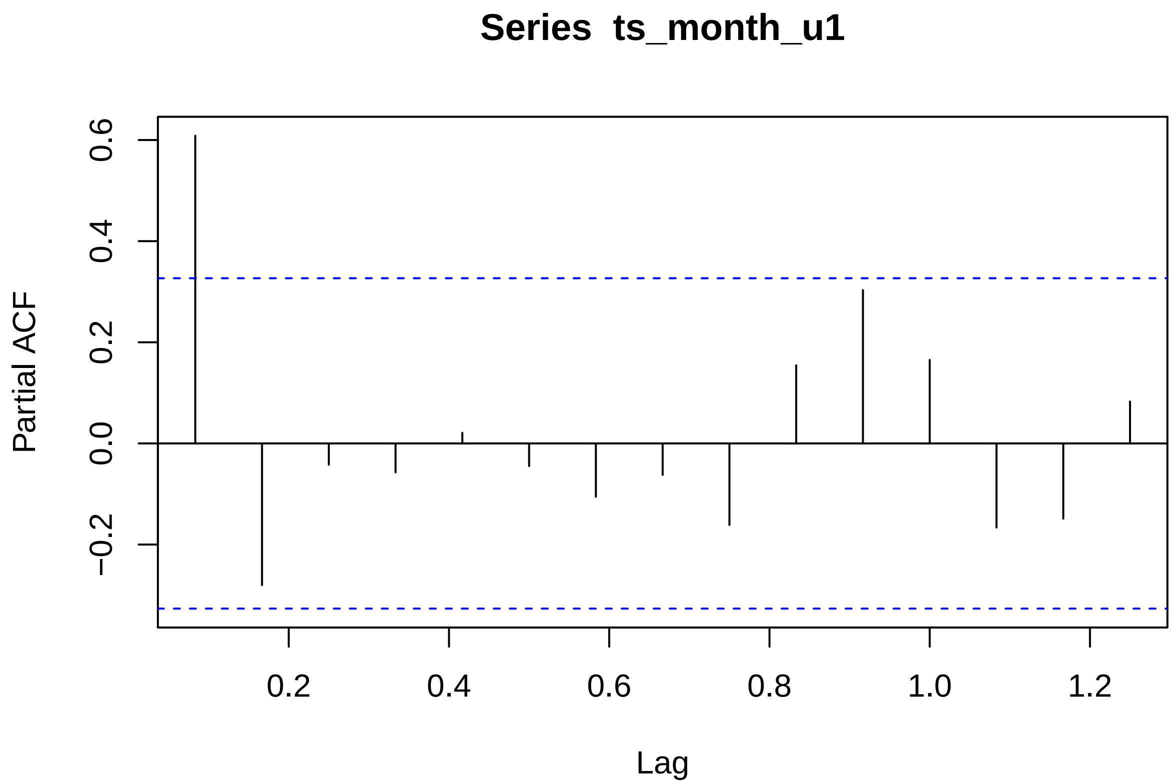
```r
checkresiduals(arima(ts_month_u6))
```



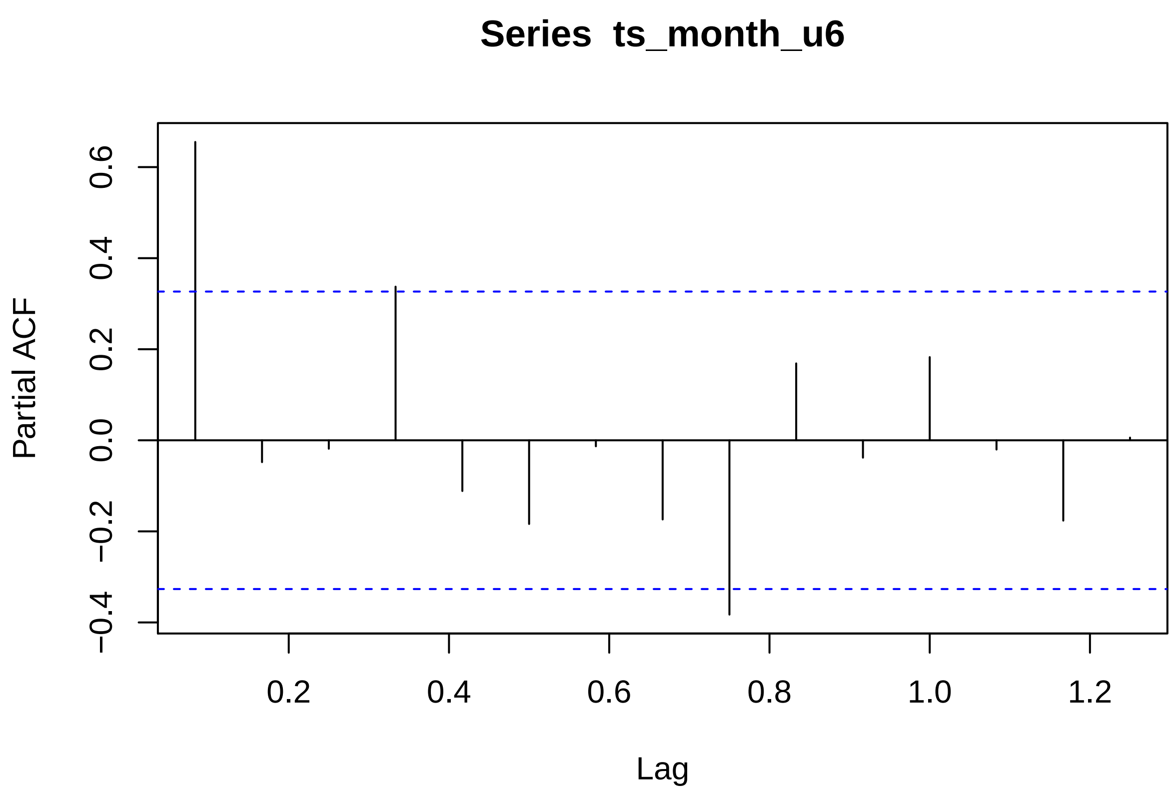

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0) with non-zero mean
## Q* = 34.506, df = 6, p-value = 5.37e-06
##
## Model df: 1.   Total lags used: 7
```

Figura 6: Residuals Analysis U6

```r
pacf(ts_month_u1)
```




```
pacf(ts_month_u6)
```




```
ts_day_u1 <- ts(both_build_df[,paste0(tag_name, "_U1")], frequency =365, start = c(2018,1), end = c(202
ts_day_u6 <- ts(both_build_df[,paste0(tag_name, "_U6")], frequency =365, start = c(2018,1), end = c(202
```

Augmented-Dickey-Fuller Test

```
adf.test(ts_day_u1, alternative = c("stationary", "explosive"),
         k = trunc((length(x)-1)^(1/3)))
```

```
## Warning in adf.test(ts_day_u1, alternative = c("stationary", "explosive"), : p-
## value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_day_u1
## Dickey-Fuller = -10.137, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(ts_day_u6, alternative = c("stationary", "explosive"),
         k = trunc((length(x)-1)^(1/3)))
```

```
## Warning in adf.test(ts_day_u6, alternative = c("stationary", "explosive"), : p-
## value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_day_u6
## Dickey-Fuller = -12.616, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

Figura 7: Partial AutoCorreleation U1-U6

Cap 2.6 Modello ARIMA con regressione temperatura, apertura/chiusura ed Effetto Covid

Carichiamo i dati meteo e aggiungiamo le variabili aperto/chiuso e covid

```
both_build_df <- weather_data_load(both_build_df, c("18","19","20"))
both_build_df <- add_dummy_open(both_build_df)
both_build_df$COVID <- 0
both_build_df[both_build_df$DATA > "2020-03-09",]$COVID <- 1
```

```
summary(lm(both_build_df[,paste0(tag_name, "_U1")] ~ aperto + tavg + COVID, both_build_df))
```

Regressione lineare U1

```
##
## Call:
## lm(formula = both_build_df[, paste0(tag_name, "_U1")] ~ aperto +
##     tavg + COVID, data = both_build_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -86.479  -7.323   0.189   7.739  31.560
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  56.40952    1.02109   55.24   <2e-16 ***
## aperto        9.24724    0.91360   10.12   <2e-16 ***
```

```
## tavg          1.08198    0.04762   22.72   <2e-16 ***
## COVID       -10.66360    0.84587  -12.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.35 on 1092 degrees of freedom
## Multiple R-squared:  0.414,  Adjusted R-squared:  0.4124
## F-statistic: 257.2 on 3 and 1092 DF,  p-value: < 2.2e-16
```

```
summary(lm(both_build_df[,paste0(tag_name, "_U6")] ~ aperto + tavg + COVID, both_build_df))
```

Regressione lineare U6

```
##
## Call:
## lm(formula = both_build_df[, paste0(tag_name, "_U6")] ~ aperto +
##     tavg + COVID, data = both_build_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -108.674  -12.535    3.664   13.782   51.434
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 121.4547     1.7906   67.83  < 2e-16 ***
## aperto       27.8809     1.6021   17.40  < 2e-16 ***
## tavg         -0.3273     0.0835   -3.92  9.4e-05 ***
## COVID       -31.1357     1.4833  -20.99  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.65 on 1092 degrees of freedom
## Multiple R-squared:  0.4095, Adjusted R-squared:  0.4078
## F-statistic: 252.4 on 3 and 1092 DF,  p-value: < 2.2e-16
```

```
M1 <- arima(ts_day_u1, order = c(3,1,1), xreg = both_build_df[, c("tavg", "aperto", "COVID")], seasonal
```

```
summary(M1)
```

Modello arima con regressione U1

```
##
## Call:
## arima(x = ts_day_u1, order = c(3, 1, 1), seasonal = list(order = c(0, 1, 0),
##     period = 365), xreg = both_build_df[, c("tavg", "aperto", "COVID")], method = "ML")
##
## Coefficients:
##          ar1      ar2     ar3      ma1    tavg  aperto    COVID
##       0.7095  -0.1347  0.1239  -0.9858  0.8675  2.7130  -6.6192
## s.e.  0.0377   0.0450  0.0374   0.0072  0.1535  0.4711   5.2669
##
## sigma^2 estimated as 109:  log likelihood = -2752.41,  aic = 5520.83
##
```

```
## Training set error measures:
##                     ME     RMSE      MAE  MPE MAPE      MASE         ACF1
## Training set 0.2295233 8.518873 4.169251 -Inf  Inf 0.9120608 -0.004677097
```

```r
M2 <- auto.arima(ts_day_u1)
```

```r
summary(M2)
```

Modello auto arima U1

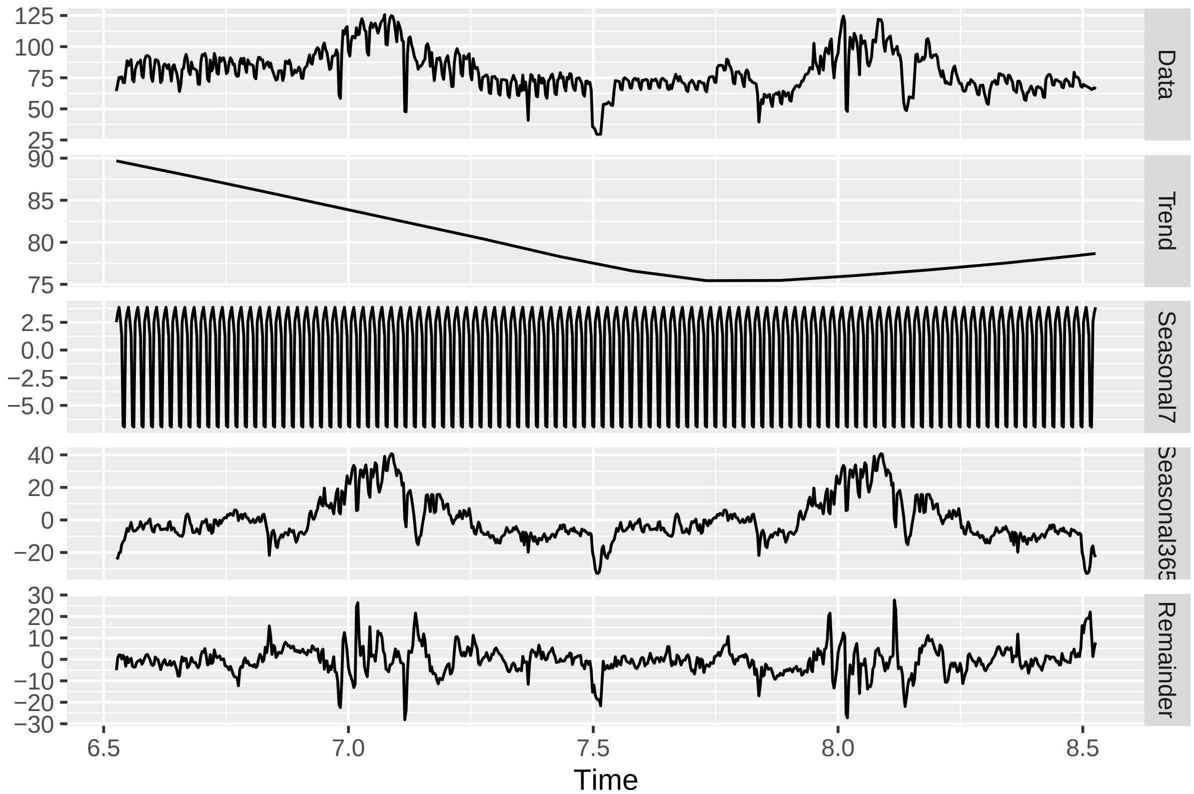
```
## Series: ts_day_u1
## ARIMA(3,1,1)(0,1,0)[365]
##
## Coefficients:
##          ar1      ar2     ar3      ma1
##       0.7148  -0.1246  0.1250  -0.9858
## s.e.  0.0381   0.0450  0.0378   0.0098
##
## sigma^2 estimated as 120.1:  log likelihood=-2785.87
## AIC=5581.74   AICc=5581.82   BIC=5604.71
##
## Training set error measures:
##                      ME     RMSE      MAE  MPE MAPE      MASE         ACF1
## Training set 0.08563434 8.918806 4.505938 -Inf  Inf 0.3605619 -0.003558162
```

Cap 2.8 Decomposizione e ricomposizione della serie con MSTL

```r
u1 <- single_build_energy_table(build = "U1", year = c("18", "19", "20"))
u1 <- time_aggregate(u1, "Day")
u1 <- u1[u1$DATA<"2020-01-02",]
u1$Target_Column <- u1$CONSUMO_ATTIVA_PRELEVATA_AVG
tag_name <- "CONSUMO_ATTIVA_PRELEVATA_AVG"
```

Decomposizione U1:

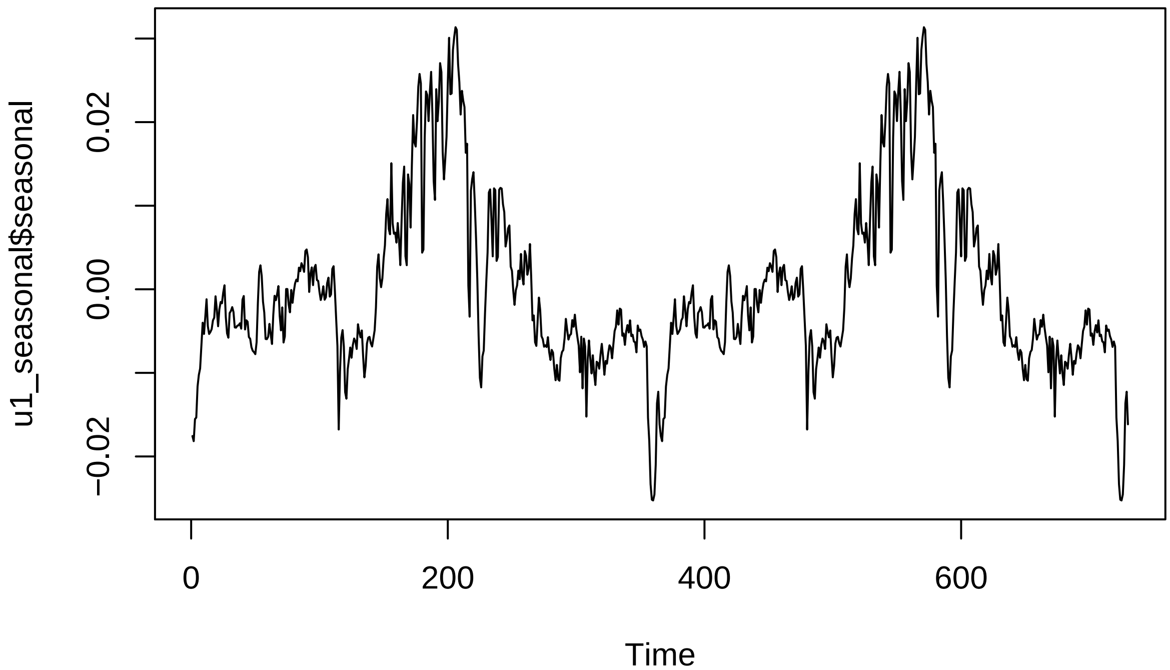
```r
u1ts <- msts(u1$Target_Column, ts.frequency = 365, start = c(01,2018), seasonal.periods = c(7,365))
u1ts_dec<-mstl(u1ts, s.window = "periodic")
u1ts_dec %>% autoplot()
```




```
rm(u1ts)
```

Figura 9 -Decomposizione multi stagionale U1 – 2018-2019

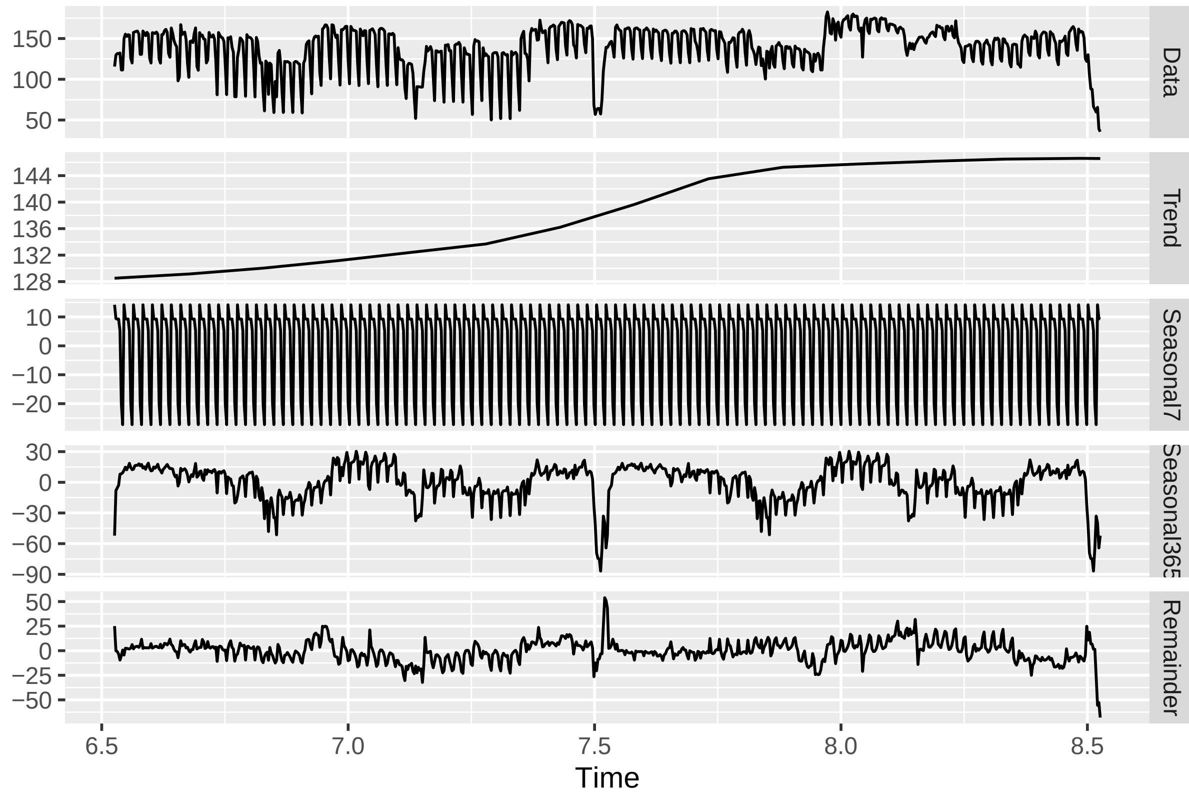
```
u1ts_comp <- as.data.frame(u1ts_dec)
u1_seasonal365 <- u1ts_comp$Seasonal365
u1_seasonal365 <- as.data.frame(u1_seasonal365)
u1_seasonal365$stand_seasonal <- u1_seasonal365$u1_seasonal365/1300

u1_seasonal <- data.frame(date=seq(from = as.Date("2018-01-01"), by = "day", length.out = 730), seasonal

plot.ts(u1_seasonal$seasonal)
```




```
rm(u1ts_dec, u1ts_comp, u1_seasonal365)
```

Decomposizione U6:

```
u6 <- u6_df
u6 <- u6[u6$DATA<"2020-01-02",]
u6$Target_Column <- u6$CONSUMO_ATTIVA_PRELEVATA_AVG

u6ts<-msts(u6$Target_Column, ts.frequency = 365, start = c(01,2018), seasonal.periods = c(7,365))
u6ts_dec<-mstl(u6ts, s.window = "periodic")
u6ts_dec %>% autoplot()
```




```
u6ts_comp <- as.data.frame(u6ts_dec)
u6ts_comp$date <- seq(from = as.Date("2018-01-01"), by = "day", length.out = nrow(u6ts_comp))
rm(u6ts, u6ts_dec)
```

Figura 9 -Decomposizione multi stagionale U6 – 2018-2019

Selezioniamo i periodi di interesse per la stagionalità annuale di U1 e U6:

```
u1_seasonal$is_hot <- rep(0, nrow(u1_seasonal))

u1_seasonal[(u1_seasonal$date >= "2018-05-01" & u1_seasonal$date<"2018-10-01"),]$is_hot <- 1

u1_seasonal[(u1_seasonal$date >= "2019-05-01" & u1_seasonal$date<"2019-10-01"),]$is_hot <- 1

u1_seasonal$seasonal_active <- u1_seasonal$is_hot * u1_seasonal$seasonal
```

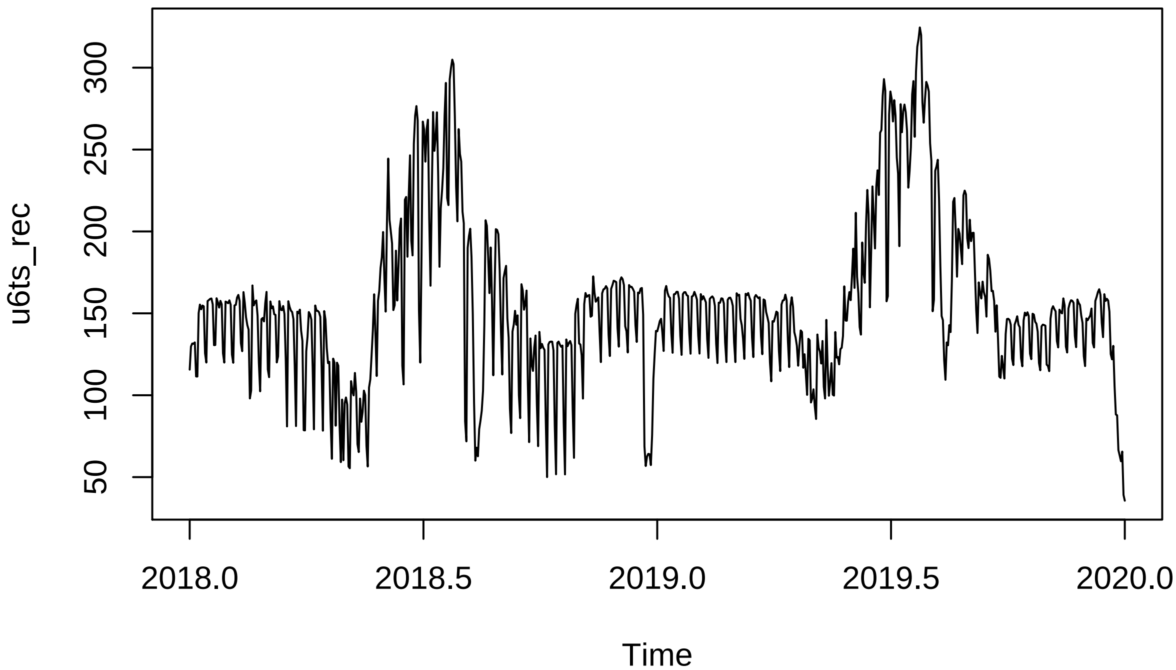
```
u6ts_comp$is_hot <- rep(1, nrow(u6ts_comp))

u6ts_comp[(u6ts_comp$date >= "2018-05-01" & u6ts_comp$date<"2018-10-01"),]$is_hot <- 0

u6ts_comp[(u6ts_comp$date >= "2019-05-01" & u6ts_comp$date<"2019-10-01"),]$is_hot <- 0

u6ts_comp$Seasonal365_Active <- u6ts_comp$Seasonal365 * u6ts_comp$is_hot
```

Ricomponiamo la serie U6 con stagionalità estiva U1:

```
u6ts_rec <- u6ts_comp$Seasonal7 + u6ts_comp$Trend+ u6ts_comp$Remainder+  (u1_seasonal$seasonal_active*53
```

```
## Warning in u6ts_comp$Seasonal7 + u6ts_comp$Trend + u6ts_comp$Remainder + :
## longer object length is not a multiple of shorter object length
```

```
u6ts_rec<-ts(u6ts_rec, frequency = 365, start = c(2018,1), end = c(2020,1))
plot.ts(u6ts_rec)
```




```
rm(u6ts_comp, u1_seasonal)
```

Figura 10 - Serie storica U6 trasformata

Calcoliamo ora i consumi effettivi con e senza teleraffreddamento.

```
monthDays <- c(31,28,31,30,31,30,31,31,30,31,30,31)
u6_rec <- data.frame(CONS=as.matrix(u6ts_rec), DATA=time(u6ts_rec))
u6_rec_consumo <- 0
for (m in 1:nrow(u6_rec)) {
  u6_rec_consumo <- u6_rec_consumo + (24*u6_rec[m,]$CONS)
}
print(paste("Consumi senza teleraffreddamento:", paste(as.character(u6_rec_consumo), "kWh")))
```

```
## [1] "Consumi senza teleraffreddamento: 2714987.25791116 kWh"
```

```
rm(monthDays, u6_rec, u6ts_rec)
```

```
monthDays <- c(31,28,31,30,31,30,31,31,30,31,30,31)
u6_consumo <- 0
for (m in 1:nrow(u6)) {
  u6_consumo <- u6_consumo + (24*u6[m,]$Target_Column)
}
print(paste("Consumi con teleraffreddamento:", paste(as.character(u6_consumo), "kWh")))
```

```
## [1] "Consumi con teleraffreddamento: 2429663.76184782 kWh"
```

```
print(paste("Risparmio in euro in due anni:", paste(as.character(round((u6_rec_consumo-u6_consumo)*0.48
```

```
## [1] "Risparmio in euro in due anni: 136955.278 €"
```

```r
print(paste("Risparmio in euro per anno:", paste(as.character(round((u6_rec_consumo-u6_consumo)*0.48/2,
```

```
## [1] "Risparmio in euro per anno: 68477.639 €"
```

```r
rm(u6,u1,u6_rec_consumo,u6_consumo)
```

Cap 2.9 Metodo ARIMA U1 e confronto

```r
train <- window(ts_day_u1, end=c(2020,2))
h <-  length(ts_day_u1) - length(train)
M3 <- auto.arima(train, lambda=1, biasadj=TRUE)
ARIMA_U1 <- forecast(M3, h=h)
```

```r
autoplot(ARIMA_U1)
```

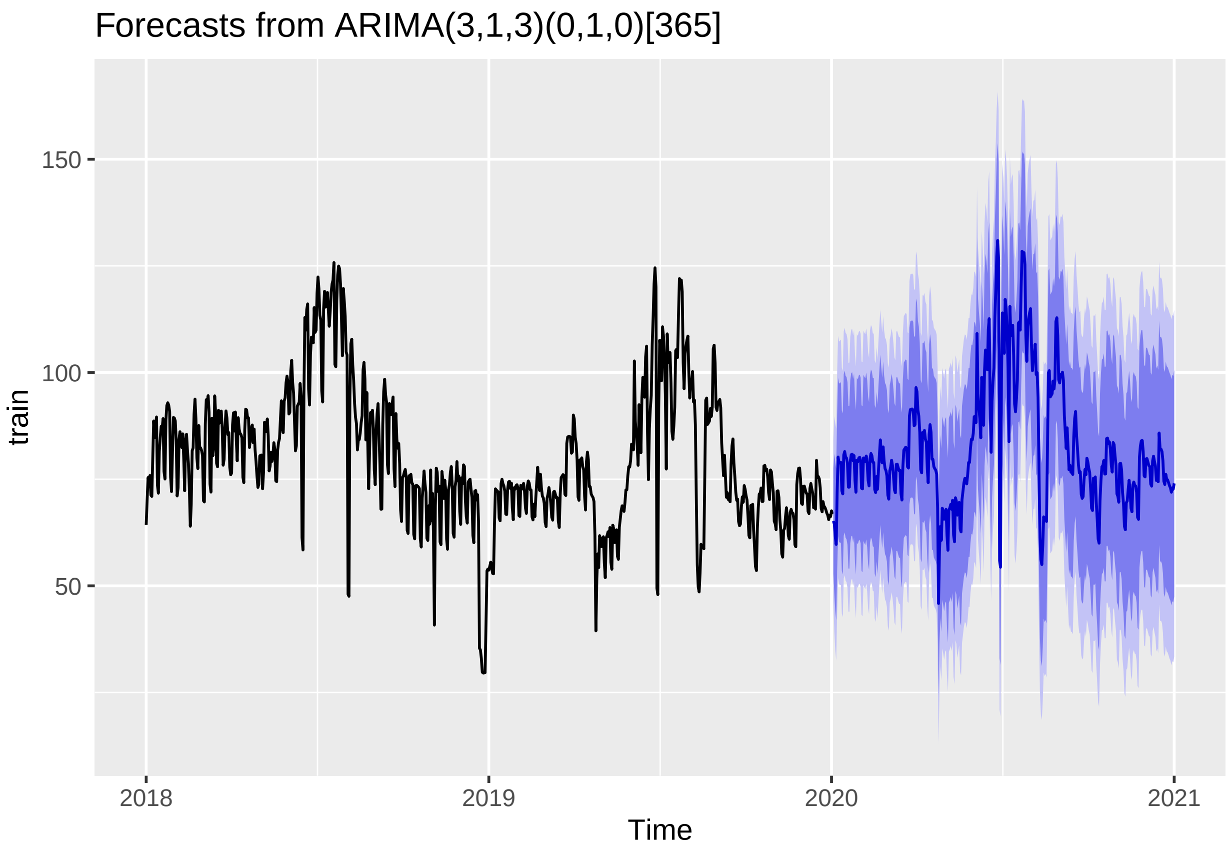


Figura 12: Bande di confidenza forecast U1

```r
autoplot(ts_day_u1) +
  autolayer(ARIMA_U1, series="ARIMA", PI=FALSE) +
  xlab("Year") + ylab(" kW") +
  ggtitle("Confronto ARIMA_U1 con e senza pandemia")
```

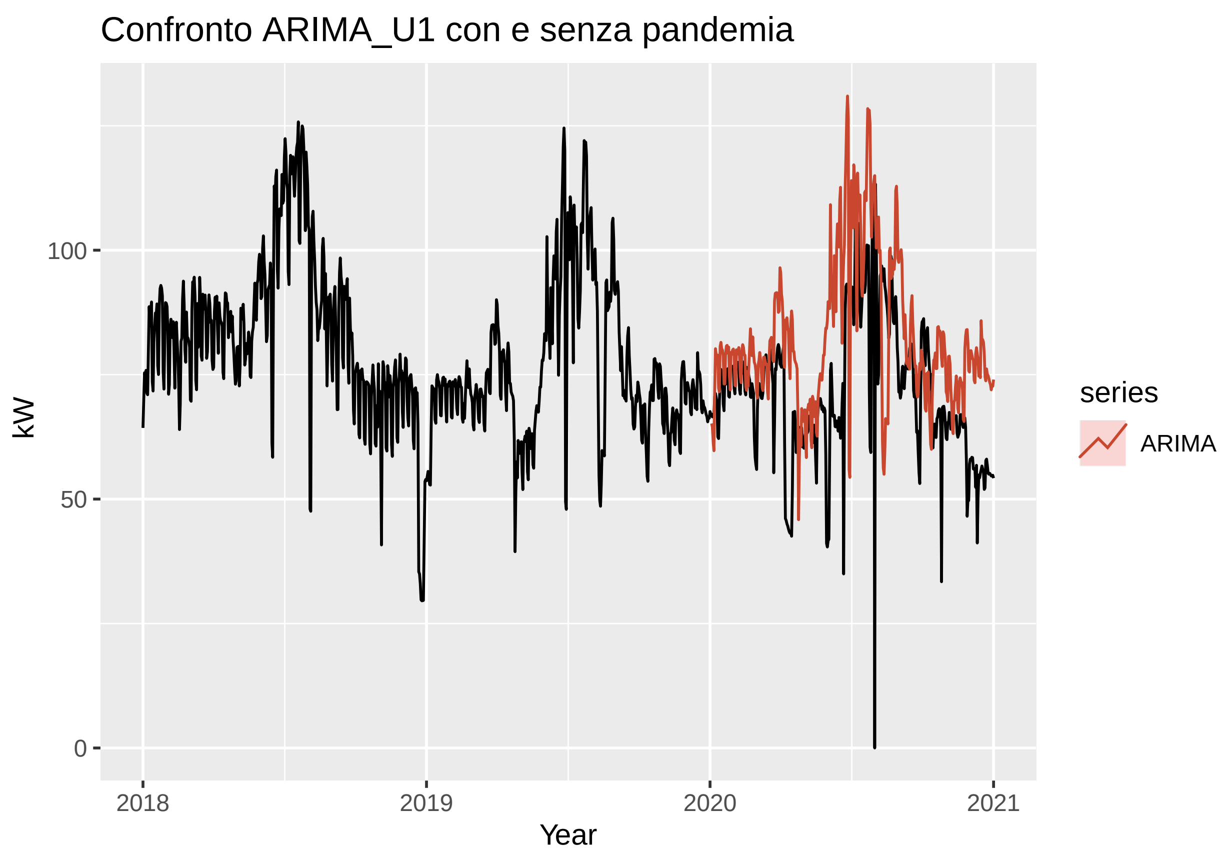


Figura 13: Confronto Covid - Non Covid U1

Cap 2.10 Metodo NEURALE U6 e confronto

```
set.seed(1234)
train <- window(ts_day_u6, end=c(2020,2))
h <-  length(ts_day_u6) - length(train)
NNAR_U6 <- forecast(nnetar(train), h=h)

autoplot(ts_day_u6) +
  autolayer(NNAR_U6, series="NNAR", PI=FALSE) +
  xlab("Year") + ylab(" kW") +
  ggtitle("Confronto NN_U6 con e senza pandemia")
```

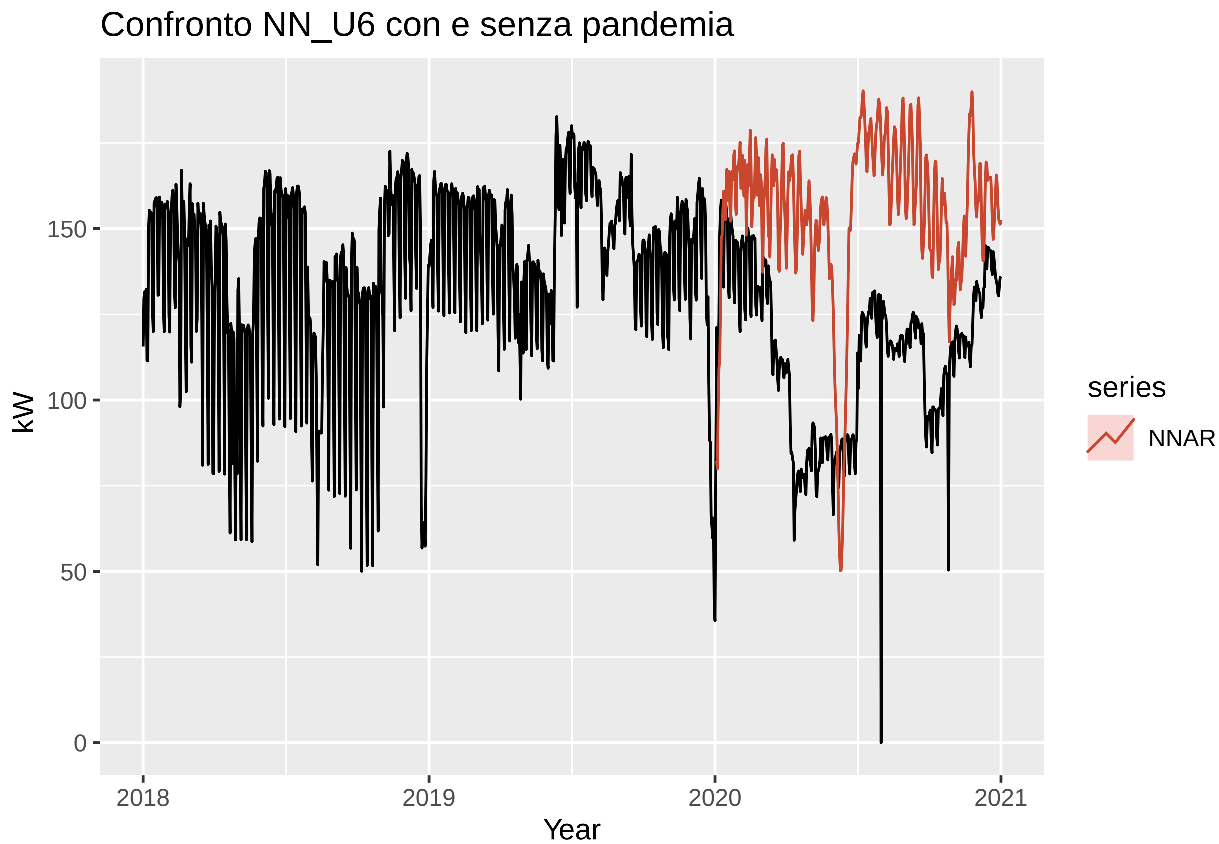


Figura 14: Confronto Covid - Non Covid U6

```
set.seed(1234)
train <- window(ts_day_u1, end=c(2020,2))
h <-  length(ts_day_u1) - length(train)
NNAR_U1 <- forecast(nnetar(train), h=h)

autoplot(ts_day_u1) +
  autolayer(NNAR_U1, series="NNAR", PI=FALSE) +
  xlab("Year") + ylab(" Confronto NN_U1 con e senza pandemia") +
  ggtitle("NN")
```

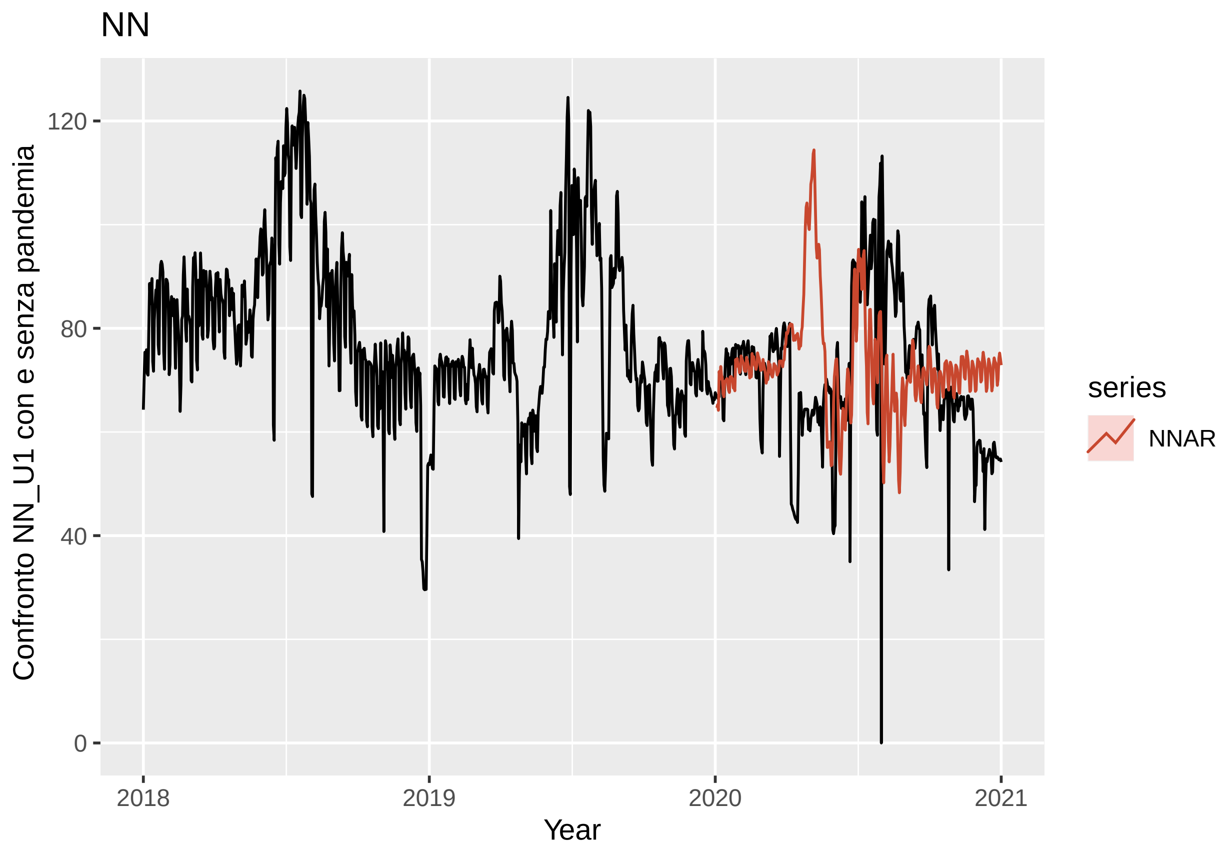


Figura 15: Confronto Covid - Non Covid U1