Untitled

2023-03-01

```
library(car)
```

```
Loading required package: carData
```

```
library(readr)
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2
--

v ggplot2 3.4.0     v dplyr   1.1.0
v tibble  3.1.8     v stringr 1.5.0
v tidyr   1.3.0     v forcats 1.0.0
v purrr   1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
x dplyr::recode() masks car::recode()
x purrr::some()   masks car::some()
```

```
library(ARDL)
```

```
To cite the ARDL package in publications:

Use this reference to refer to the validity of the ARDL package.

  Natsiopoulos, Kleanthis, and Tzeremes, Nickolaos G. (2022). ARDL
  bounds test for cointegration: Replicating the Pesaran et al. (2001)
  results for the UK earnings equation using R. Journal of Applied
  Econometrics, 37(5), 1079-1090. https://doi.org/10.1002/jae.2919

Use this reference to cite this specific version of the ARDL package.

  Kleanthis Natsiopoulos and Nickolaos Tzeremes (2023). ARDL: ARDL, ECM
  and Bounds-Test for Cointegration. R package version 0.2.2.
  https://CRAN.R-project.org/package=ARDL
```

```
library(xts)
```

```
Loading required package: zoo
```

```
Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric


Attaching package: 'xts'

The following objects are masked from 'package:dplyr':

    first, last
```

```
library(car)
library(xtable)
library(ISOweek)

# LOAD DATA #

eex <- read_csv("data/eex.csv", col_types = cols(Datum = col_date(format = "%d.%m.%Y"))) %>%
  rename(date = Datum, frontjahr = `Frontjahr, Euro/MWh`, price_day_ahead = `Day-Ahead (1 MW), Euro/MWh`

demand <- read_csv("data/gastag.csv", col_types = cols(Gastag = col_date(format = "%d.%m.%Y")))%>%
  rename(date = Gastag, demand = `Restlast[kWh]*`)


weather_cloud <- read_csv("data/cloud.csv", col_names = TRUE, cols("Date" = col_date(format = "%Y-%m-%d
weather <- read_csv("data/hdd16.csv", col_names = TRUE, cols("Date" = col_date(format = "%Y-%m-%d"))) %>
  mutate(HDD16 = HDD16+ 0.00000001)

weather_wind10 <- read_csv("data/wind.csv", col_names = TRUE, cols("Date" = col_date(format = "%Y-%m-%d

weighted_weather <- read_csv("data/bl_weighted_weather.csv",
                             col_types = cols(date = col_date(format = "%Y-%m-%d")))


weighted_weather
```

```
# A tibble: 1,876 x 7
   date       wind_speed_sum temperature_air_mea~1 tempe~2 tempe~3 celct~4 hdd16
   <date>              <dbl>                 <dbl>   <dbl>   <dbl>   <dbl> <dbl>
 1 2018-01-01           6.15                  279.    277.    282.    7.26  8.74
 2 2018-01-02           4.89                  278.    276.    279.    5.43 10.6
 3 2018-01-03           8.46                  279.    277.    282.    6.92  9.08
 4 2018-01-04           5.81                  280.    278.    283.    7.57  8.43
 5 2018-01-05           5.03                  280.    278.    282.    7.73  8.27
 6 2018-01-06           2.58                  278.    276.    280.    6.01  9.99
 7 2018-01-07           4.73                  275.    273.    277.    2.93 13.1
 8 2018-01-08           5.21                  275.    273.    277.    2.39 13.6
 9 2018-01-09           4.04                  276.    273.    279.    4.16 11.8
10 2018-01-10           2.47                  278.    275.    281.    5.43 10.6
# ... with 1,866 more rows, and abbreviated variable names
#   1: temperature_air_mean_200_sum, 2: temperature_air_min_200_sum,
#   3: temperature_air_max_200_sum, 4: celctemp
```

```r
#weighted_weather_pop <- read_csv("data/bl_weighted_weather (1).csv",
#                                 col_types = cols(date = col_date(format = "%Y-%m-%d")))

google_trends <- read_csv("data/google_trend_indicator.csv",
                                                    col_types = cols(...1 = col_skip(), date = col_date
```

```
New names:
* `` -> `...1`
```

```r
# --- Joining --- #

#daily_freq <- inner_join(demand, weather, by=c("date"="Date")) %>%
daily_freq <- inner_join(demand, weighted_weather, by=c("date")) %>%
  inner_join(eex, by=c("date")) %>%

  #rename variables
  rename(wind=wind_speed_sum) %>%
  dplyr::select(date, hdd16, wind, demand, price_day_ahead) %>%

  #log transforms
  mutate(demand = demand /1000,
         log_demand = log(demand),
         log_day_ahead = log(price_day_ahead),
         eex_lag = lag(price_day_ahead))

#Demeaning per week
demdata <- daily_freq %>%
  mutate(week = substr(as.character(ISOweek(date)),7,8)) %>%
  group_by(week) %>%
  mutate(demand = demand - mean(demand), hdd16 = hdd16 - mean(hdd16))



#More variables
demdata$eex_prewar= ifelse(demdata$date < as.Date("2022-02-24"), demdata$price_day_ahead, 0)
demdata$war_d = ifelse(demdata$date >= as.Date("2022-02-24"), 1, 0)
demdata$eex_postwar = ifelse(demdata$date >= as.Date("2022-02-24"), demdata$price_day_ahead, 0)
demdata$covid = ifelse(demdata$date >= as.Date("2020-02-24"), demdata$price_day_ahead, 0)

# --- ARDL --- #

model <- auto_ardl(demand ~ hdd16 + wind + price_day_ahead, data = demdata, max_order = 5, selection =

summary(model$best_model)
```

```
Time series regression with "ts" data:
Start = 6, End = 1302

Call:
dynlm::dynlm(formula = full_formula, data = data, start = start,
    end = end)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-464669  -24941     254   27097  408446

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -1.297e+04  5.468e+03  -2.372 0.017816 *
L(demand, 1)     8.475e-01  1.477e-02  57.376  < 2e-16 ***
hdd16            7.192e+04  9.633e+02  74.667  < 2e-16 ***
L(hdd16, 1)     -4.701e+04  1.647e+03 -28.544  < 2e-16 ***
L(hdd16, 2)     -5.280e+03  1.280e+03  -4.125 3.95e-05 ***
L(hdd16, 3)     -2.405e+03  1.266e+03  -1.900 0.057682 .
L(hdd16, 4)      8.335e+02  1.253e+03   0.665 0.505988
L(hdd16, 5)     -1.403e+03  9.553e+02  -1.469 0.142152
wind             1.820e+04  1.639e+03  11.102  < 2e-16 ***
L(wind, 1)      -1.288e+04  1.663e+03  -7.746 1.91e-14 ***
price_day_ahead -1.174e+02  3.357e+01  -3.497 0.000487 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 57950 on 1286 degrees of freedom
Multiple R-squared:  0.9481,    Adjusted R-squared:  0.9477
F-statistic:  2348 on 10 and 1286 DF,  p-value: < 2.2e-16
```

```
ml = multipliers(model$best_model)
ms = multipliers(model$best_model, "sr")

ms
```

```
             Term    Estimate Std. Error   t value     Pr(>|t|)
1     (Intercept) -12972.5061  5467.9436 -2.372465 1.781625e-02
2           hdd16  71924.1263   963.2611 74.667322 0.000000e+00
3            wind  18197.9209  1639.2143 11.101612 2.071848e-27
4 price_day_ahead   -117.4093    33.5746 -3.496967 4.866137e-04
```

```
ml
```

```
             Term    Estimate Std. Error   t value     Pr(>|t|)
1     (Intercept) -85058.9385 36325.7910 -2.341558 1.935546e-02
2           hdd16 109238.4521  5760.5209 18.963294 6.616238e-71
3            wind  34854.4349  9696.5031  3.594537 3.372588e-04
4 price_day_ahead   -769.8366   209.8802 -3.667981 2.544582e-04
```

```
avg_price <- mean(daily_freq$price_day_ahead)
avg_demand <- mean(daily_freq$demand)
avg_hdd <- mean(daily_freq$hdd16)
avg_wind <- mean(daily_freq$wind)

#price
ms$Estimate[4]*avg_price/avg_demand
```

```
[1] -0.004772241
```

```
ml$Estimate[4]*avg_price/avg_demand
```

```
[1] -0.03129093
```

```
#price postwar
ms$Estimate[5]*avg_price/avg_demand
```

```
[1] NA
```

```
ml$Estimate[5]*avg_price/avg_demand
```

```
[1] NA
```

```
ms
```

```
             Term    Estimate Std. Error   t value     Pr(>|t|)
1     (Intercept) -12972.5061  5467.9436 -2.372465 1.781625e-02
2           hdd16  71924.1263   963.2611 74.667322 0.000000e+00
3            wind  18197.9209  1639.2143 11.101612 2.071848e-27
4 price_day_ahead   -117.4093    33.5746 -3.496967 4.866137e-04
```

```
#hdd
ms$Estimate[2]*avg_hdd/avg_demand
```

```
[1] 0.3898736
```

```
ml$Estimate[2]*avg_hdd/avg_demand
```

```
[1] 0.5921406
```

```
#wind
ms$Estimate[3]*avg_wind/avg_demand
```

```
[1] 0.05658603
```

```
ml$Estimate[3]*avg_wind/avg_demand
```

```
[1] 0.1083791
```

```
#Model with two regimes


cbind(ms, ml)
```

```
             Term    Estimate Std. Error   t value     Pr(>|t|)            Term
1     (Intercept) -12972.5061  5467.9436 -2.372465 1.781625e-02     (Intercept)
2           hdd16  71924.1263   963.2611 74.667322 0.000000e+00           hdd16
3            wind  18197.9209  1639.2143 11.101612 2.071848e-27            wind
4 price_day_ahead   -117.4093    33.5746 -3.496967 4.866137e-04 price_day_ahead
     Estimate Std. Error   t value     Pr(>|t|)
1 -85058.9385 36325.7910 -2.341558 1.935546e-02
2 109238.4521  5760.5209 18.963294 6.616238e-71
3  34854.4349  9696.5031  3.594537 3.372588e-04
4   -769.8366   209.8802 -3.667981 2.544582e-04
```

```
model_t <- auto_ardl(demand ~ hdd16 + wind + eex_prewar + eex_postwar, data = demdata, max_order = 5, s

summary(model_t$best_model)
```

```
Time series regression with "ts" data:
Start = 6, End = 1302

Call:
dynlm::dynlm(formula = full_formula, data = data, start = start,
    end = end)

Residuals:
    Min      1Q  Median      3Q     Max
-454569  -23791     485   28790  401087

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.887e+04  5.554e+03  -3.399 0.000698 ***
L(demand, 1)  7.764e-01  2.729e-02  28.454  < 2e-16 ***
L(demand, 2)  4.946e-02  2.676e-02   1.848 0.064778 .
hdd16         7.193e+04  9.565e+02  75.202  < 2e-16 ***
L(hdd16, 1)  -4.210e+04  2.271e+03 -18.533  < 2e-16 ***
L(hdd16, 2)  -7.799e+03  2.031e+03  -3.841 0.000129 ***
L(hdd16, 3)  -2.591e+03  1.275e+03  -2.032 0.042321 *
L(hdd16, 4)   7.458e+02  1.246e+03   0.599 0.549532
L(hdd16, 5)  -1.403e+03  9.497e+02  -1.477 0.139868
wind          1.776e+04  1.629e+03  10.906  < 2e-16 ***
L(wind, 1)   -1.226e+04  1.658e+03  -7.397 2.51e-13 ***
eex_prewar    1.689e+02  7.266e+01   2.325 0.020237 *
eex_postwar  -1.465e+02  3.437e+01  -4.261 2.18e-05 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 57450 on 1284 degrees of freedom
Multiple R-squared:  0.949, Adjusted R-squared:  0.9486
F-statistic:  1993 on 12 and 1284 DF,  p-value: < 2.2e-16
```

```
ml = multipliers(model_t$best_model)
ms = multipliers(model_t$best_model, "sr")
#price
ms$Estimate[4]*avg_price/avg_demand
```

```
[1] 0.006866058
```

```
ml$Estimate[4]*avg_price/avg_demand
```

```
[1] 0.03943192
```

```
#price postwar
ms$Estimate[5]*avg_price/avg_demand
```

```
[1] -0.005954041
```

```
ml$Estimate[5]*avg_price/avg_demand
```

```
[1] -0.03419419
```

```
#hdd
ms$Estimate[2]*avg_hdd/avg_demand
```

```
[1] 0.3899121
```

```
ml$Estimate[2]*avg_hdd/avg_demand
```

```
[1] 0.5848385
```

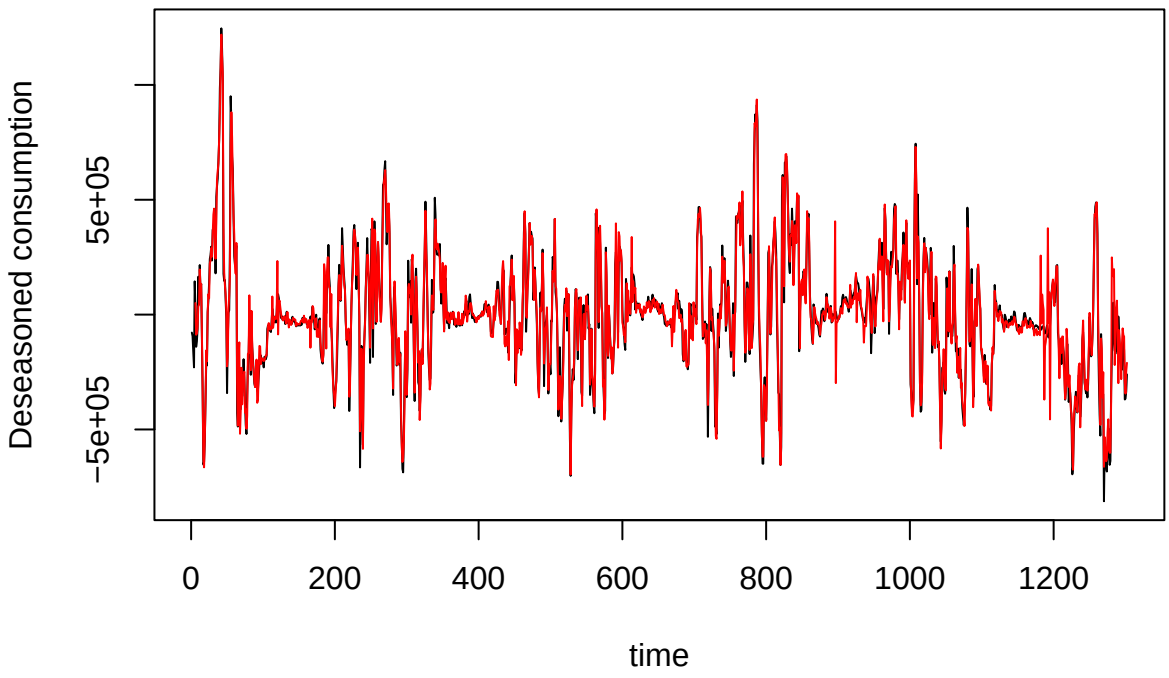
```
#wind
ms$Estimate[3]*avg_wind/avg_demand
```

```
[1] 0.05523511
```

```
ml$Estimate[3]*avg_wind/avg_demand
```

```
[1] 0.09821655
```

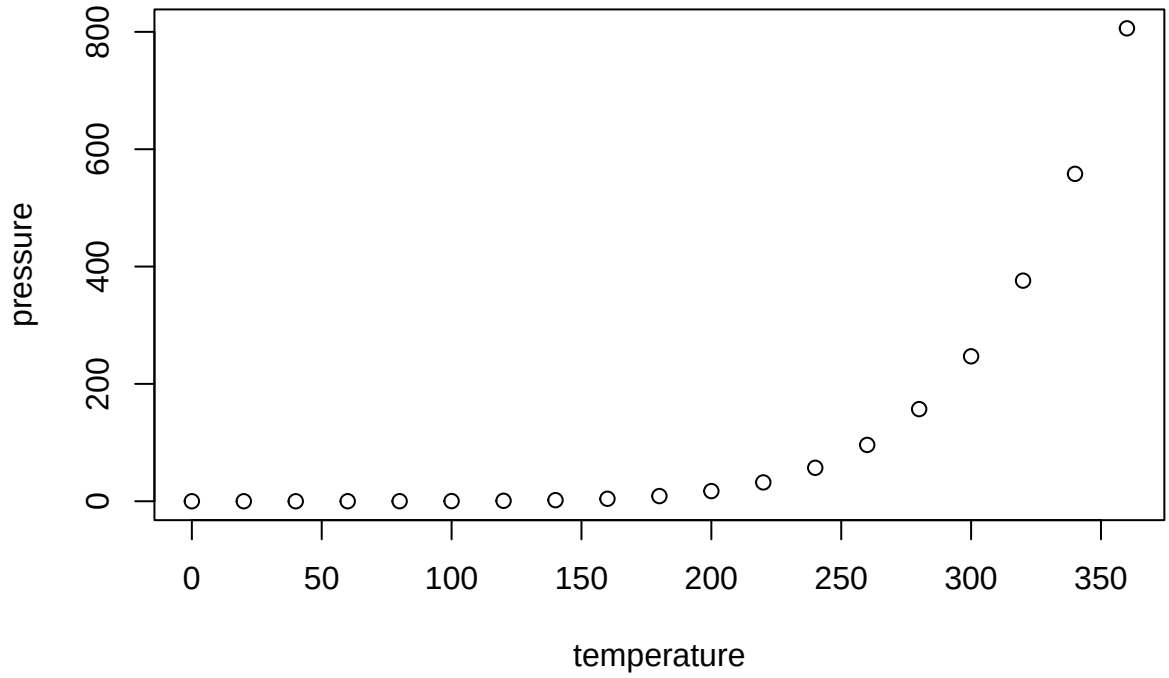
```
plot(demdata$demand, type="l", ylab="Deseasoned consumption", xlab="time")
plot(demdata$demand, type="l", ylab="Deseasoned consumption", xlab="time")
lines(model$best_model$fitted.values, col="red")
```




```
try = demdata %>% group_by(format(date, "%m-%Y")) %>% summarise(price = mean(price_day_ahead))
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.