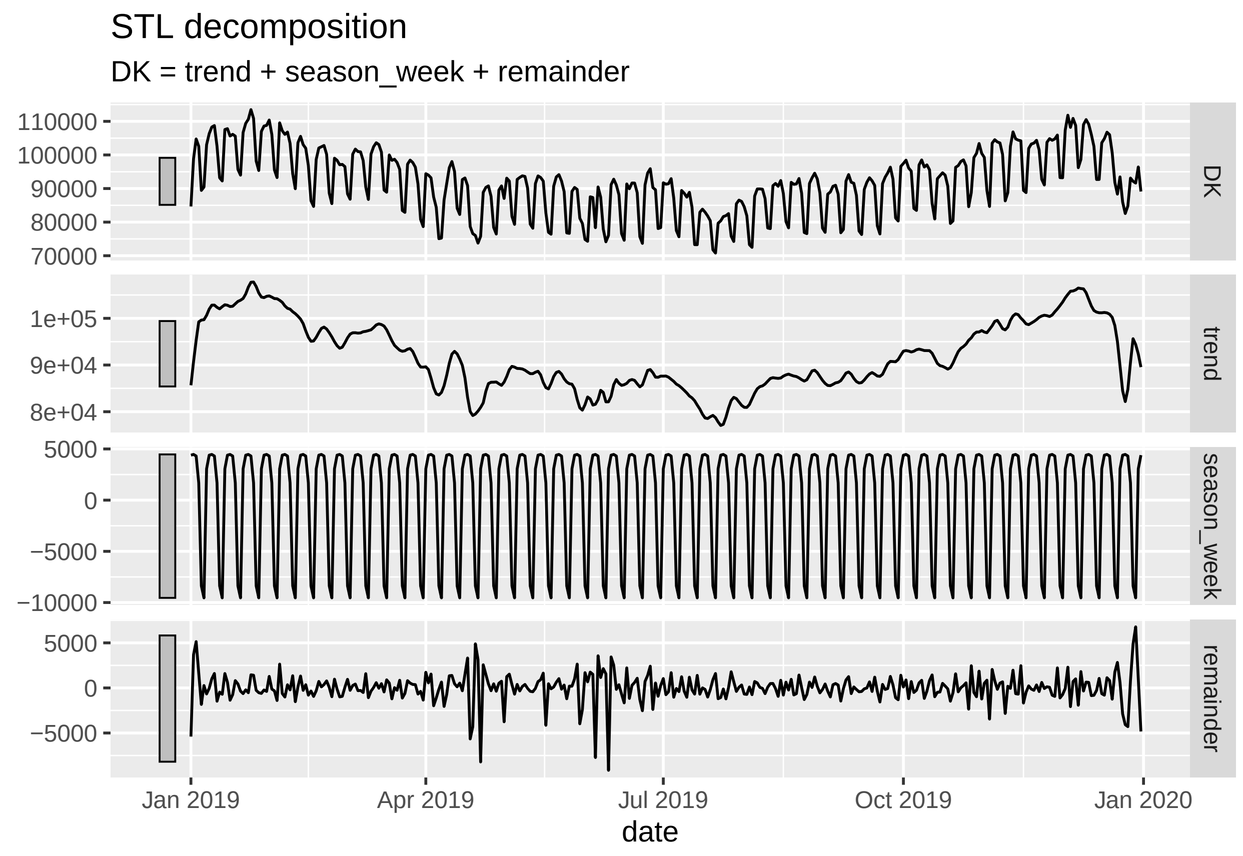
Assignment 5

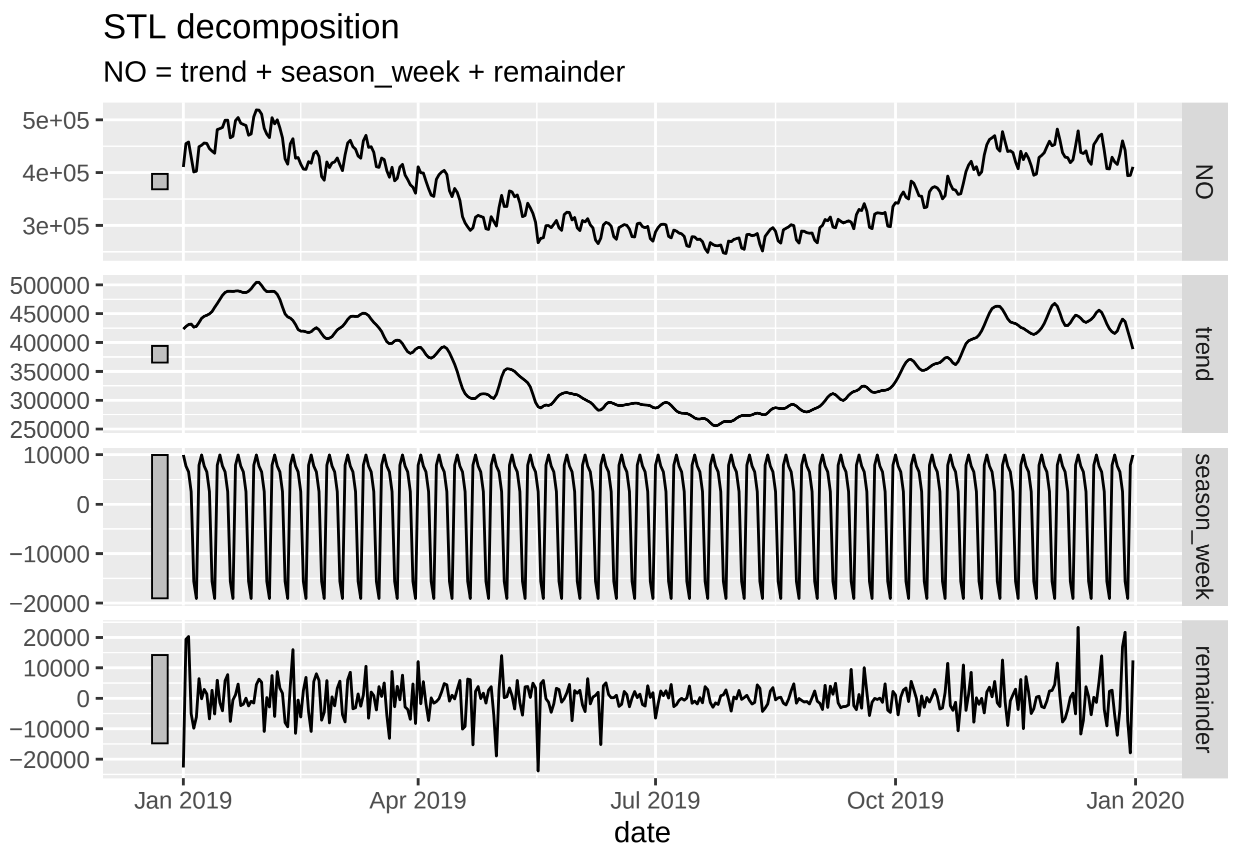
Task 1: Comparison of consumption data in Norway and Denmark

We start our comparison of norwegian and danish power consumption by looking at the STL decompositon plots.

```
## Seasonal decompositon of Danish electricity consumption

cons_comp_dk = cons_ts %>% model(
  STL(DK ~ trend(window=7) + season(window="periodic"))
) %>% components

cons_comp_no = cons_ts %>% model(
  STL(NO ~ trend(window=7) + season(window="periodic"))
) %>% components


cons_comp_dk %>%  autoplot()
```



```
cons_comp_no %>%  autoplot()
```



From the components we can see the same general trend of winter increase, and corresponding the decrease in power consumption in the summer months. Where they differ is the magnitude of the weekly-sesaonal component. In Denmark the weekend effect of reduced power consumption is more noticable.

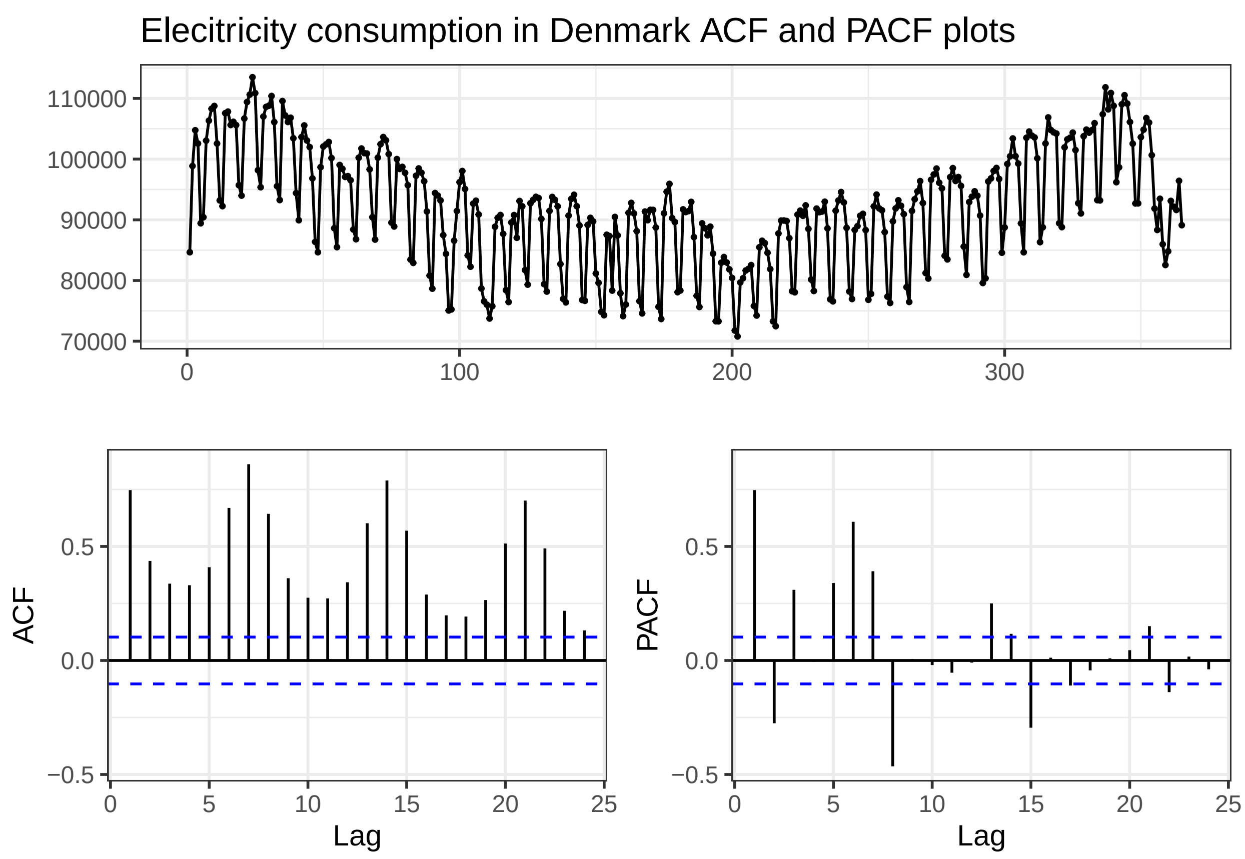
Another small difference is the effect of winter, where it appears that the winter-increase in power consumption is larger in Norway than Denmark.

Forecasting using a seasonal ARIMA model

A clear assumption in an ARIMA forecasting model is the that the data is stationary in terms of its variance and mean. We plot the time series containing electricity consumption data, as well its autocorrelation and partial autocorrelation plots. We see clear signs of non-stationarity and perform unit root tests confirming the need for differencing. As there appears to be a strong seasonal autocorrelation, we will first conduct a seasonal differencing, and see if this solves our non-stationarity issue. If not, further differencing will be needed. Some information contained in the data is lost by performing a differencing, but we conform the the assumption of stationarity of the data.

```
forecast::ggtsdisplay(cons$DK, plot_type='partial',
                      lag.max = 24,
                      theme = theme_bw(),
                      main = "Elecitricity consumption in Denmark ACF and PACF plots ")
```

```
## Warning: Ignoring unknown parameters: plot_type
```



```
unitroot_kpss(cons$DK)
```

```
##   kpss_stat kpss_pvalue
##    1.219224    0.010000
```

```
adf.test(cons$DK)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cons$DK
## Dickey-Fuller = -2.17, Lag order = 7, p-value = 0.5052
## alternative hypothesis: stationary
```

We perform a first order differencing and perform the same stationarity analysis.

```
# Perform differencing

cons_diff_dk <- cons %>% mutate(DK = difference(DK,7)) %>% dplyr::filter(!is.na(DK)) #Take first order

unitroot_kpss(cons_diff_dk$DK)
```

```
##   kpss_stat kpss_pvalue
##   0.1060349   0.1000000
```

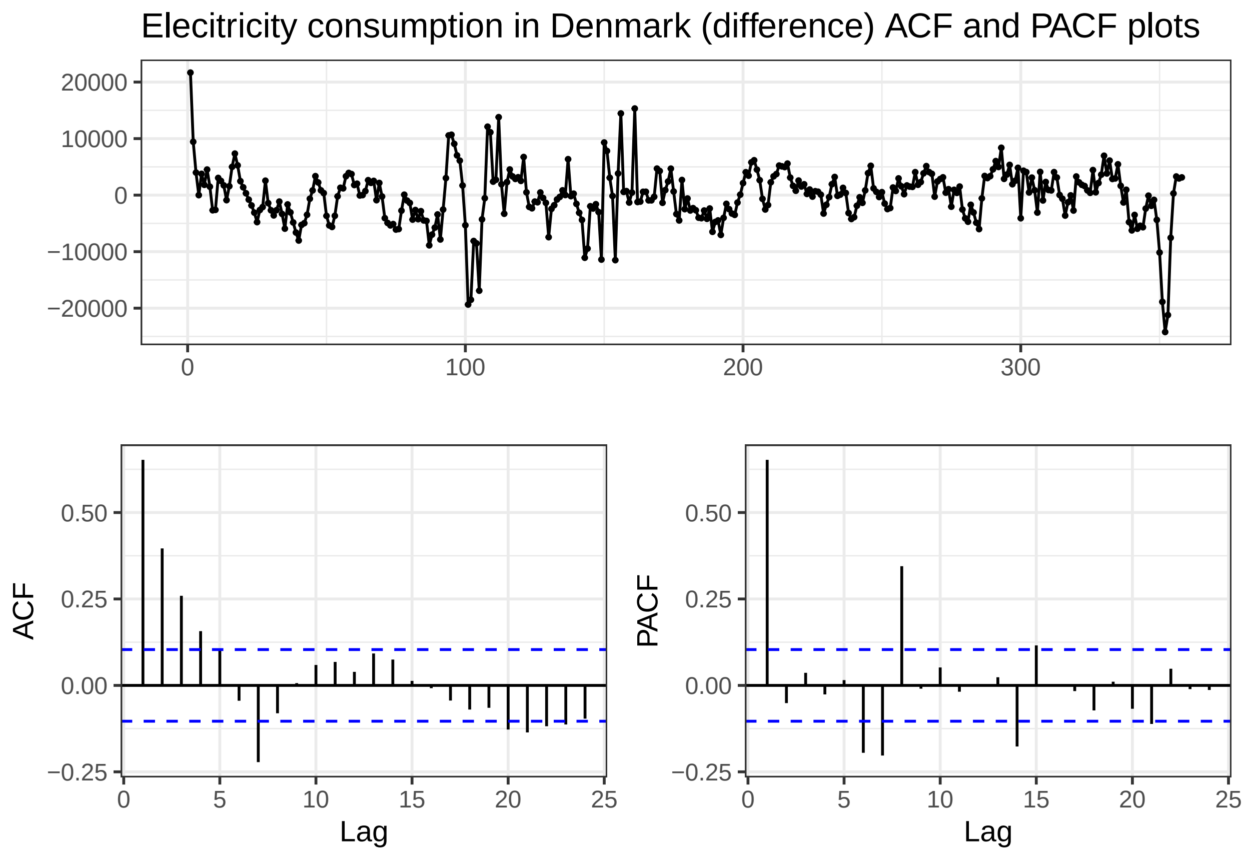
```r
adf.test(cons_diff_dk$DK) #Stationary
```

```
## Warning in adf.test(cons_diff_dk$DK): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cons_diff_dk$DK
## Dickey-Fuller = -5.5696, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# New ACF and PACF plots

forecast::ggtsdisplay(cons_diff_dk$DK, plot_type='partial',
                      lag.max = 24,
                      theme = theme_bw(),
                      main = "Elecitricity consumption in Denmark (difference) ACF and PACF plots ")
```

```
## Warning: Ignoring unknown parameters: plot_type
```



We note that there are significant autocorrelations at the weekly lag (i.e 7, 14). Luckily the fable package correctly identified the seasonality as weekly, e.g an ARIMA PDQ pdq[7], regardless of the specific terms.

We will now perform two forecasts, a manually specified ARIMA model and an automatically determined ARIMA model made by the fable ARIMA() function. In the PACF plot we can see a clear autocorrelation in seasonal lag terms, in a decreasing fashion. This calls for an MA(1) term to applied to the seasonal component of the ARIMA model. There is a significant but decreasing correlation at lag 1 as shown in the PACF plot, and an AR(1) term applied to the non-seasonal component is appropiate. As the lags are decreasing after 1, and MA(1) term might also be necessary.

A such we might reason that our model might something like: ARIMA pdq(1,0,1) pdq(1,1,1)

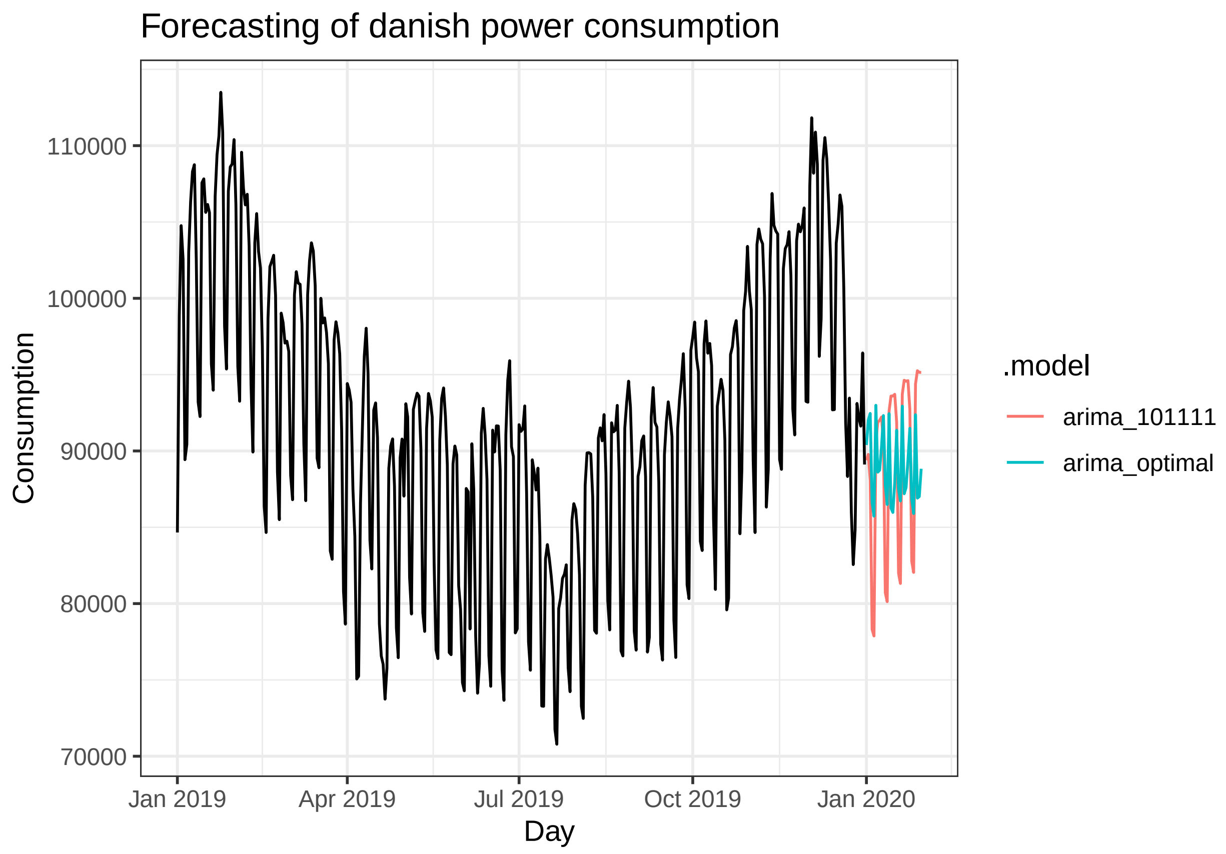
```
fit_arima_optimal_cons_dk <-
  cons %>%
  as_tsibble(index = date) %>%
  model(arima_optimal =  ARIMA(DK, stepwise = FALSE, approximation = FALSE))


fit_arima_manual_cons_dk <- cons %>% as_tsibble(index = date) %>%
  model(arima_101111       = ARIMA(DK ~ 0 + pdq(1,0,1) + PDQ(1,1,1)))

fit_cons_dk  <-  fit_arima_manual_cons_dk  %>% bind_cols(fit_arima_optimal_cons_dk)
```

In the plot below we have used our manually selected model to forecast danish power consumption for a time horizon of 30 days.

```
#Forecast for h = 30
fc_cons_dk <-  fit_cons_dk %>% forecast(h = 30)


fc_cons_dk  %>% ggplot() +
  geom_line(aes(x = date, y = .mean, color = .model)) +
  geom_line(aes(x  = date, y = DK), data = cons) +
  theme_bw()   +
   labs(title = "Forecasting of danish power consumption",
        y = "Consumption",
        x = "Day")
```

Table 1: Training data performance metrics: Danish power consumption

Model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	AC
arima_101111	Training	-48.38959	2860.688	1853.999	-0.1144917	2.052392	0.5298189	0.5781630	-0.03580
arima_optimal	Training	-50.35888	3070.275	2111.181	-0.1114627	2.331917	0.6033140	0.6205218	-0.02976



As we can see our manually selected model slightly outperforms the optimally selected model based on AIC. The Ljung Box reveals that there is no residual autocorrelation and as such much of the error term is explained in our model.

```
# Forecast evaluation

fit_cons_dk %>%  accuracy()  %>%
  rename("Model" = .model) %>%
  kbl(caption = "Training data performance metrics: Danish power consumption") %>%
  kable_classic(full_width = F, html_font = "Times new roman")
```

```
# Autocorrelation tests

fit_cons_dk  %>%
     augment()  %>% dplyr::filter(.model == "arima_101111")
```

```
## # A tsibble: 365 x 6 [1D]
## # Key:       .model [1]
##    .model       date           DK .fitted  .resid  .innov
##    <chr>        <date>      <dbl>   <dbl>   <dbl>   <dbl>
##  1 arima_101111 2019-01-01  84659  84574.    84.7    84.7
##  2 arima_101111 2019-01-02  98864  98765.    98.9    98.9
##  3 arima_101111 2019-01-03 104761 104656.   105.    105.
##  4 arima_101111 2019-01-04 102569 102466.   103.    103.
##  5 arima_101111 2019-01-05  89429  89340.    89.4    89.4
##  6 arima_101111 2019-01-06  90429  90339.    90.4    90.4
##  7 arima_101111 2019-01-07 103042 102939.   103.    103.
##  8 arima_101111 2019-01-08 106334  97294.  9040.   9040.
##  9 arima_101111 2019-01-09 108301 114463. -6162.  -6162.
## 10 arima_101111 2019-01-10 108753 110536. -1783.  -1783.
## # ... with 355 more rows
```

```
feasts::ljung_box(
  (fit_cons_dk  %>%
     augment()  %>% dplyr::filter(.model == "arima_101111"))$.innov) # Passing
```

```
##   lb_stat lb_pvalue
## 0.4718201 0.4921510
```

Perhaps the most notable weakness of such a model is its weekly sesaonality. It would be interesting to tweak the model with longer seasonality, as a week may be too small a period to capture the winter/summer differences in consumption.

Another weakness is that some information useful for forecasting may be found in other variables, and as a result a multivariate model should also be tested.

Task 2:

We retrieve finnish electricity price data from Nordpool and clean them.

```
elspot_data <- read_csv("elspot-prices_2019_daily_nok.csv") # Load data
```

```
## Warning: Missing column names filled in: 'X2' [2], 'X3' [3], 'X4' [4], 'X5' [5],
## 'X6' [6], 'X7' [7], 'X8' [8], 'X9' [9], 'X10' [10], 'X11' [11], 'X12' [12],
## 'X13' [13], 'X14' [14], 'X15' [15], 'X16' [16], 'X17' [17], 'X18' [18],
## 'X19' [19], 'X20' [20], 'X21' [21], 'X22' [22], 'X23' [23]
```

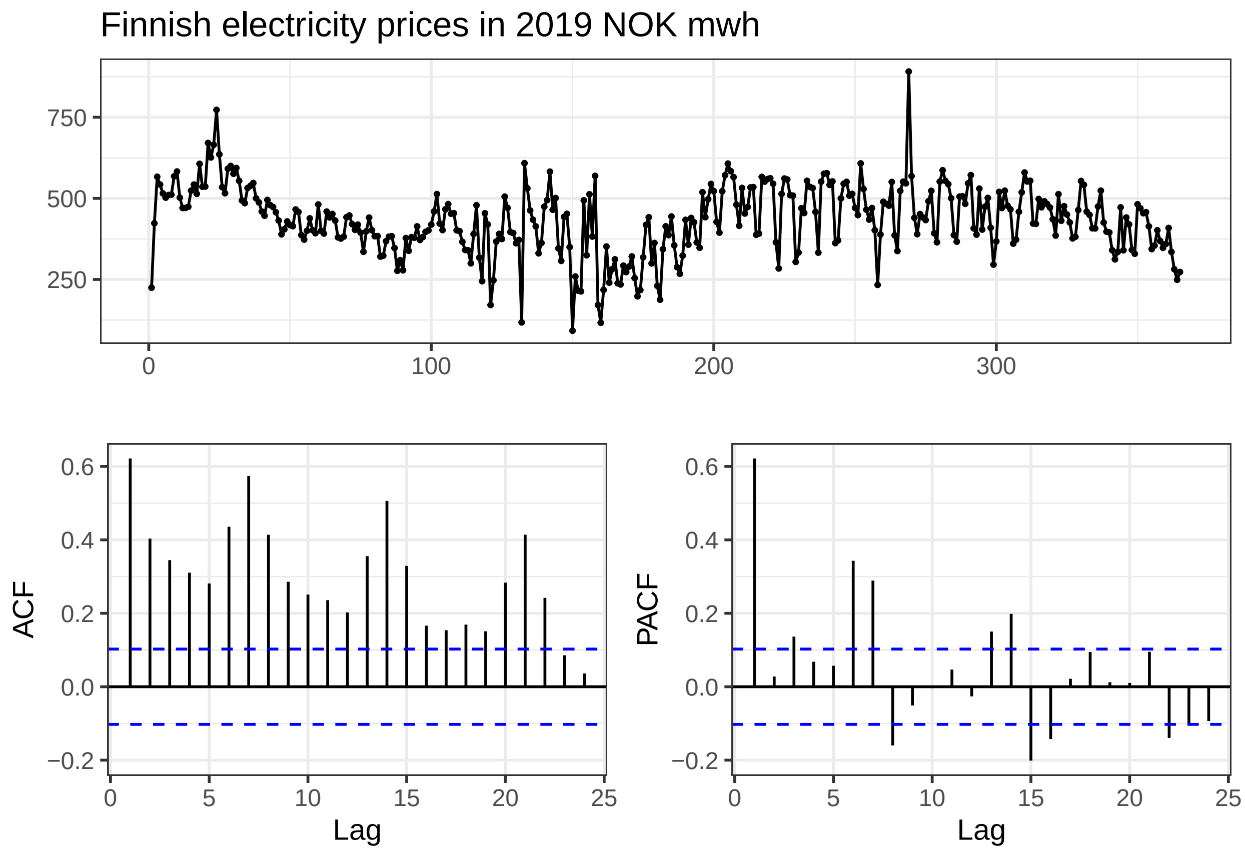
```
## Parsed with column specification:
## cols(
##   .default = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
colnames(elspot_data) <- elspot_data[2,]
elspot_data <- elspot_data[3:nrow(elspot_data),]
colnames(elspot_data)[1] <- "date"

elspot_data  %<>%
```

```
    mutate(date = lubridate::dmy(date),
           FI = as.numeric(gsub(",", ".", FI)))  %>%  #Substitute commas
    select(date, FI)  %>%
    as_tsibble(index = date)

#Plot of finnish electricity prices and ACF/PACF plots
forecast::ggtsdisplay(elspot_data$FI, plot_type='partial',
                      lag.max = 24,
                      theme = theme_bw(),
                      main = "Finnish electricity prices in 2019 NOK mwh")
```

```
## Warning: Ignoring unknown parameters: plot_type
```



We find the conditional variance of the series by calculating the residuals after an ARIMA model (in our case an ARIMA(1,1) model). By subtracting the original values with with the fitted values of this ARIMA model, we end up with the "return residuals".

```
fi_garch <- garchFit(~arma(1,1) + garch(1,1), data = elspot_data$FI, trace = F)
```

```
## Warning: Using formula(x) is deprecated when x is a character vector of length > 1.
##   Consider formula(paste(x, collapse = " ")) instead.
```

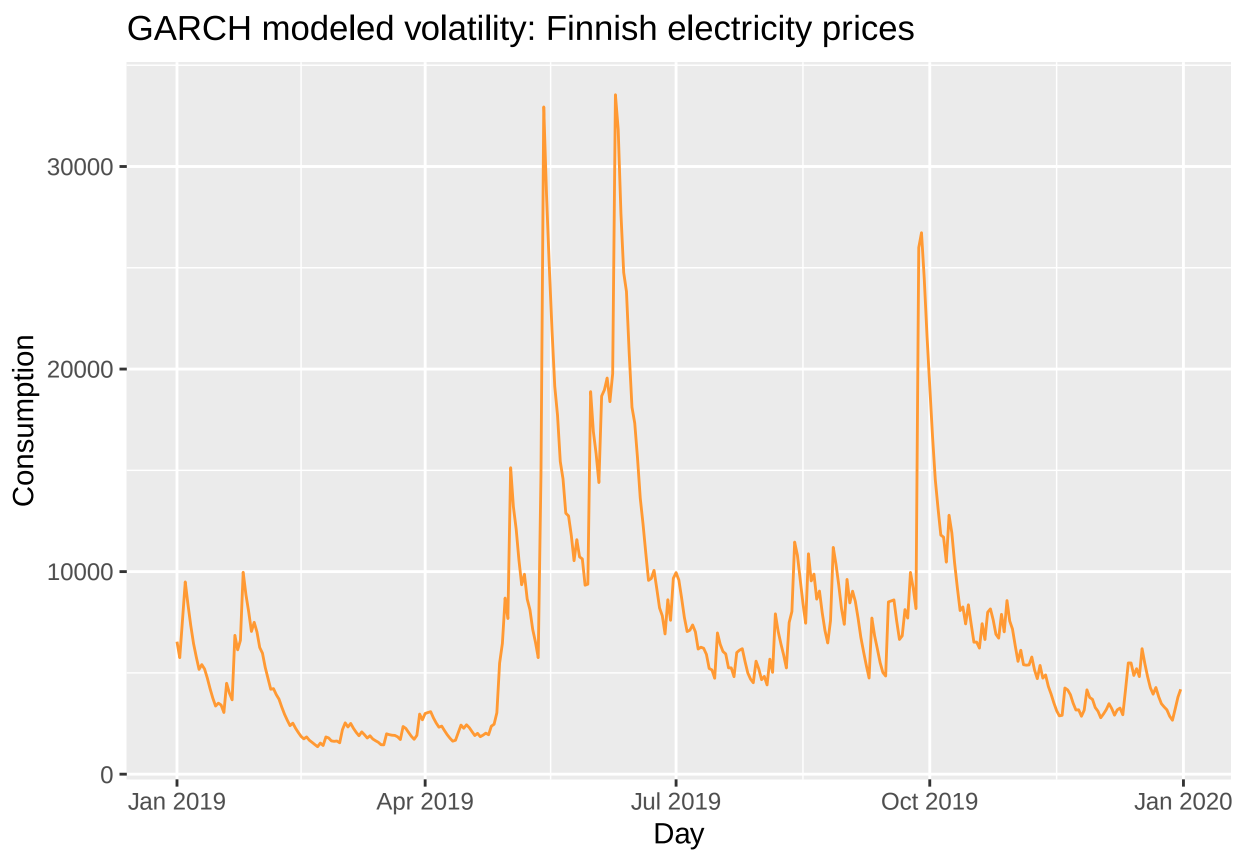
```
summary(fi_garch)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(1, 1) + garch(1, 1), data = elspot_data$FI,
##     trace = F)
##
## Mean and Variance Equation:
##  data ~ arma(1, 1) + garch(1, 1)
## <environment: 0x00000000132beb80>
##  [data = elspot_data$FI]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        mu        ar1        ma1      omega     alpha1      beta1
## 123.56463    0.71407   -0.03790   81.57607    0.13143    0.86680
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     123.56463    33.04677    3.739 0.000185 ***
## ar1      0.71407     0.07565    9.439  < 2e-16 ***
## ma1     -0.03790     0.12114   -0.313 0.754387
## omega   81.57607    63.61225    1.282 0.199704
## alpha1   0.13143     0.03585    3.666 0.000246 ***
## beta1    0.86680     0.03105   27.918  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  -2088.787    normalized:  -5.722703
##
## Description:
##  Mon Apr 12 21:52:03 2021 by user: sondr
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  25.00763  3.712463e-06
##  Shapiro-Wilk Test  R    W      0.9874976 0.00312319
##  Ljung-Box Test     R    Q(10)  97.34201  2.220446e-16
##  Ljung-Box Test     R    Q(15)  189.0253  0
##  Ljung-Box Test     R    Q(20)  213.1669  0
##  Ljung-Box Test     R^2  Q(10)  11.40592  0.3267794
##  Ljung-Box Test     R^2  Q(15)  18.17036  0.2538154
##  Ljung-Box Test     R^2  Q(20)  23.56921  0.2617165
##  LM Arch Test       R    TR^2   14.24749  0.285184
##
## Information Criterion Statistics:
##      AIC      BIC      SIC     HQIC
## 11.47828 11.54239 11.47775 11.50376
```

We note the signifiance of the alpha and gamma parameters, and our time series variance appears to be conditional on previous variance.

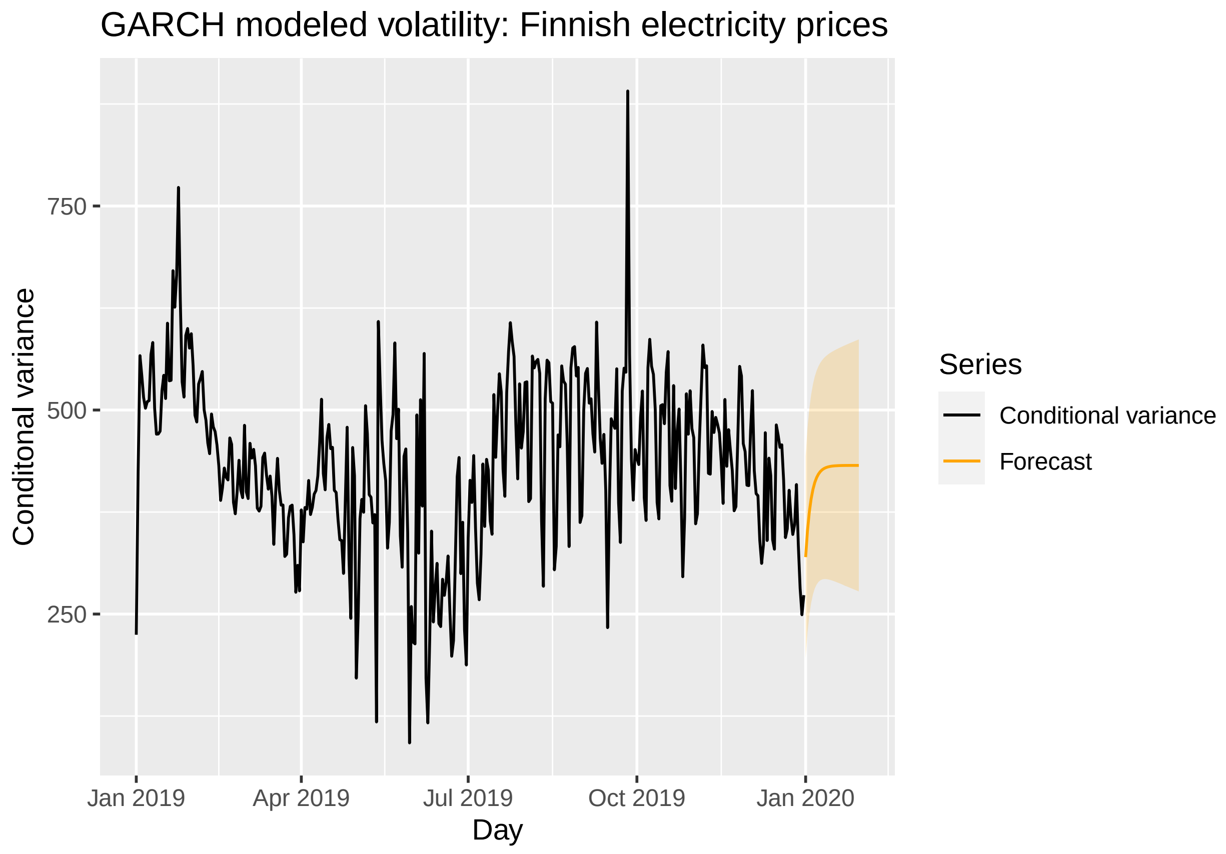
We model the conditonal volatility of the time series, and note that there are several significant spikes in volatility of electricity prices.

```
elspot_data  %<>% mutate(volatility = fi_garch@h.t)
elspot_data  %>%
    ggplot(aes(y = volatility, x = date)) +
    geom_line(col = '#ff9933')  +
    labs(title = "GARCH modeled volatility: Finnish electricity prices",
         y = "Consumption",
         x = "Day")
```



```
fc_garch = predict(fi_garch, n.ahead = 30)
fc_garch["date"] = seq(as.Date("2020-01-01"), as.Date("2020-01-01")+29, by = "days")



elspot_data  %>%
```

```r
ggplot() +
geom_line(aes(x = date, y = FI, color = "Conditional variance")) +
geom_line(aes(x = date, y = meanForecast, color = "Forecast"), data =  fc_garch) +
geom_ribbon(aes(x=date, ymin=(meanForecast-2*standardDeviation), ymax=(meanForecast+2*standardDeviati
scale_colour_manual(values = c("black", "orange")) +
labs(title = "GARCH modeled volatility: Finnish electricity prices",
       y = "Conditonal variance",
       x = "Day",
       colour = "Series")
```



Task 3