**Ivan Berlim Gonçalves**

---

## 1 - BUSINESS PROBLEM

We want to predict the `OILPRICE` values given the oil dataset from GAMLSS package.

The price of oil and other commodities is influenced by three major factors: supply, demand and geopolitics. But what if we could indirectly predict its price using other commodities and stock indexes?

This is the proposed challenge that we are going to solve. I'm going to make a superficial scratch on the complex world of natural resource pricing.

Great! We already have our problem question:

- What will be the tomorrow's oil price?

We have a data set containing different prices indexes that hopefully will help predict the response variable `OILPRICE` which represents the log price of front month WTI oil contract traded by NYMEX.

Now let us establish some tangible objectives to what we want as an answer. Given that the stock market is a complex, volatile and fast environment, we must set a goal for assessing the model's metrics. A variation of 2% in price prediction could mean a lot of wasted money depending on the amount applied. So we are going to set two high standards goals for our model performance:

- at least 90% of the predictions have to be on a 1% deviance margin from the real values.
- at least 75% of the predictions have to be on a 0.5% deviance margin from the real values.

We are ready to start. It's time to understand the data.

---

## 2 - UNDERSTANDING THE DATA

**What is the GAMLSS package?** Generalized Additive Models for Location, Scale and Shape (GAMLSS) were introduced by Rigby and Stasinopoulos (2001, 2005) and Akantziliotou et al. (2002) as a way of overcoming some of the limitations associated with Generalized Linear Models (GLM) and Generalized Additive Models (GAM) (Nelder and Wedderburn, 1972 and Hastie and Tibshirani, 1990, respectively).

In GAMLSS the exponential family distribution assumption for the response variable (Y) is relaxed and replaced by a general distribution family, including highly skew and/or kurtotic distributions. The systematic part of the model is expanded to allow modeling not only the mean (or location) but other parameters of the distribution of Y as linear parametric and/or additive non-parametric functions of explanatory variables and/or random effects. Maximum (penalized) likelihood estimation is used to fit the models.

There are two algorithms to fit the models, the CG and RS algorithms.

---

**GAMLSS guide used to this project:** http://www.gamlss.com/wp-content/uploads/2013/01/gamlss-manual.pdf

---

**Dataset Information:** Link: https://rdrr.io/cran/gamlss.data/man/oil.html

---

**Dataset Description** The Oil data: Using model selection to discover what affects the price of oil. The data s contains the daily prices of front-month WTI (West Texas Intermediate) oil prices traded by NYMEX (New York Mercantile Exchange). The front-month WTI oil price is a futures contract with the shortest duration that could be purchased in the NYMEX market. The idea is to use other financially traded products (e.g., gold price) to discover what might affect the daily dynamics of the price of oil.

---

**Dataset Source** The dataset was downloaded from https://www.quandl.com/.

---

**Dataset Format**

**OILPRICE** the log price of front month WTI oil contract traded by NYMEX - in financial terms, this is the CL1. This is the response variable.

**CL_log** numeric vectors which are the log prices of the 2 to 15 months ahead WTI oil contracts traded by NYMEX. For example, for the trading day of 2nd June 2016, the CL2 is the WTI oil contract for delivery in August 2016.

**BDIY_log** the Baltic Dry Index, which is an assessment of the price of moving the major raw materials by sea.

**SPX_log** the S&P 500 index

**DX1_log** the US Dollar Index

**GC1_log** The log price of front month gold price contract traded by NYMEX

**HO1_log** the log price of front month heating oil contract traded by NYMEX

**USCI_log** the United States Commodity Index

**GNR_log** the S&P Global Natural Resources Index

**SHCOMP_log** the Shanghai Stock Exchange Composite Index.

**FTSE_log** the FTSE 100 Index

**respLAG** the lag 1 of OILPRICE - lagged version of the response variable.

---

## 3 - DATA PREPARATION

---

```
# Verify if the package is already installed, if not, install package
if("ggplot2" %in% rownames(installed.packages()) == FALSE) {install.packages("ggplot2")}
if("ggpubr" %in% rownames(installed.packages()) == FALSE) {install.packages("ggpubr")}
if("tidyverse" %in% rownames(installed.packages()) == FALSE) {install.packages("tidyverse")}
if("tidyr" %in% rownames(installed.packages()) == FALSE) {install.packages("tidyr")}
if("gridExtra" %in% rownames(installed.packages()) == FALSE) {install.packages("gridExtra")}
if("gamlss" %in% rownames(installed.packages()) == FALSE) {install.packages("gamlss")}
if("gamlss.add" %in% rownames(installed.packages()) == FALSE) {install.packages("gamlss.add")}
if("gamlss.dist" %in% rownames(installed.packages()) == FALSE) {install.packages("gamlss.dist")}
if("corrplot" %in% rownames(installed.packages()) == FALSE) {install.packages("corrplot")}
if("gamlss.dist" %in% rownames(installed.packages()) == FALSE) {install.packages("gamlss.dist")}
if("Hmisc" %in% rownames(installed.packages()) == FALSE) {install.packages("Hmisc")}
if("forecast" %in% rownames(installed.packages()) == FALSE) {install.packages("forecast")}

# Loading libraries
library(gamlss)
library(ggpubr)
library(gamlss.add)
library(gamlss.dist)
library(ggplot2)
library(corrplot)
library(cowplot)
library(Hmisc)
library(gridExtra)
library(forecast)
```

### Packages and Libraries

---

Extract the GAMLSS Oil dataset as `oil` data frame:

```
# extrack the 'oil' data

data(oil)

# set random seed to create a reproducible script
set.seed(18)
```

---

## 4 - EXPLORATORY ANALYSIS

We want to understand how the data is organized in the data frame:

```
head(as.matrix(oil), 3)
```

```
##      OILPRICE  CL2_log  CL3_log  CL4_log  CL5_log  CL6_log  CL7_log  CL8_log
## [1,] 4.640923 4.636475 4.641116 4.644968 4.648038 4.649761 4.650908 4.651863
## [2,] 4.633077 4.645352 4.649857 4.653484 4.656338 4.657858 4.658711 4.659564
## [3,] 4.634049 4.637831 4.642466 4.646312 4.649665 4.651672 4.653103 4.654341
##       CL9_log CL10_log CL11_log CL12_log CL13_log CL14_log CL15_log BDIY_log
## [1,] 4.652340 4.651672 4.650621 4.648613 4.646120 4.643236 4.639765 6.850126
## [2,] 4.660037 4.659374 4.657952 4.655578 4.652531 4.649187 4.645256 6.850126
## [3,] 4.655293 4.655007 4.653865 4.651672 4.648804 4.645736 4.642081 6.879356
##       SPX_log  DX1_log  GC1_log  HO1_log USCI_log  GNR_log SHCOMP_log FTSE_log
## [1,] 7.221624 4.386554 7.413367 1.136197 4.108412 3.917806   7.744539 8.636699
## [2,] 7.235309 4.379762 7.419680 1.152564 4.120986 3.942552   7.762536 8.650062
## [3,] 7.222756 4.387449 7.418481 1.155182 4.115127 3.923952   7.766061 8.639729
##       respLAG
## [1,] 4.631812
## [2,] 4.640923
## [3,] 4.633077
```

---

Let's see how many rows (observations) and columns (variables) there are in the `oil` data frame:

```
paste0("The \'oil\' data frame has ", nrow(oil), " observations and  ",
       ncol(oil), " variables. ")
```

```
## [1] "The 'oil' data frame has 1000 observations and  25 variables. "
```

---

Now we want to know what are the variables' names in the `oil` data frame. They correspond to the descriptions in the `Dataset Format` section at the description of this project.

```
# columns names (oil)
oil_col_names <- colnames(oil)
cat('Variables in the \'oil\' data frame: \n\n')
```

```
## Variables in the 'oil' data frame:
```

```
for (i in (1:ncol(oil))) {print(paste0("Column number: ",i, ". Variable name: ",
                                       oil_col_names[i]))}
```

```
## [1] "Column number: 1. Variable name: OILPRICE"
## [1] "Column number: 2. Variable name: CL2_log"
## [1] "Column number: 3. Variable name: CL3_log"
## [1] "Column number: 4. Variable name: CL4_log"
## [1] "Column number: 5. Variable name: CL5_log"
```

```
## [1] "Column number: 6. Variable name: CL6_log"
## [1] "Column number: 7. Variable name: CL7_log"
## [1] "Column number: 8. Variable name: CL8_log"
## [1] "Column number: 9. Variable name: CL9_log"
## [1] "Column number: 10. Variable name: CL10_log"
## [1] "Column number: 11. Variable name: CL11_log"
## [1] "Column number: 12. Variable name: CL12_log"
## [1] "Column number: 13. Variable name: CL13_log"
## [1] "Column number: 14. Variable name: CL14_log"
## [1] "Column number: 15. Variable name: CL15_log"
## [1] "Column number: 16. Variable name: BDIY_log"
## [1] "Column number: 17. Variable name: SPX_log"
## [1] "Column number: 18. Variable name: DX1_log"
## [1] "Column number: 19. Variable name: GC1_log"
## [1] "Column number: 20. Variable name: HO1_log"
## [1] "Column number: 21. Variable name: USCI_log"
## [1] "Column number: 22. Variable name: GNR_log"
## [1] "Column number: 23. Variable name: SHCOMP_log"
## [1] "Column number: 24. Variable name: FTSE_log"
## [1] "Column number: 25. Variable name: respLAG"
```

---

Checking if there are any null values in the `oil` data frame:

```
r is.null(oil)
## [1] FALSE
r sum(is.na(oil))
## [1] 0
```

Great! There is no null value in the `oil` data frame

---

It is time to create a data frame called `df` that contains all `oil` data frame variables except the log prices of the 2 to 15 months ahead WTI oil contracts traded by NYMEX, the so-called `CL#_log` variables.

Our new `df` data frame holds only the commodities and stocks indexes, the predictor variable `OILPRICE`, and its lagged version, the `respLAG` variable.

The idea is to remove unnecessary columns that may add noise to our data:

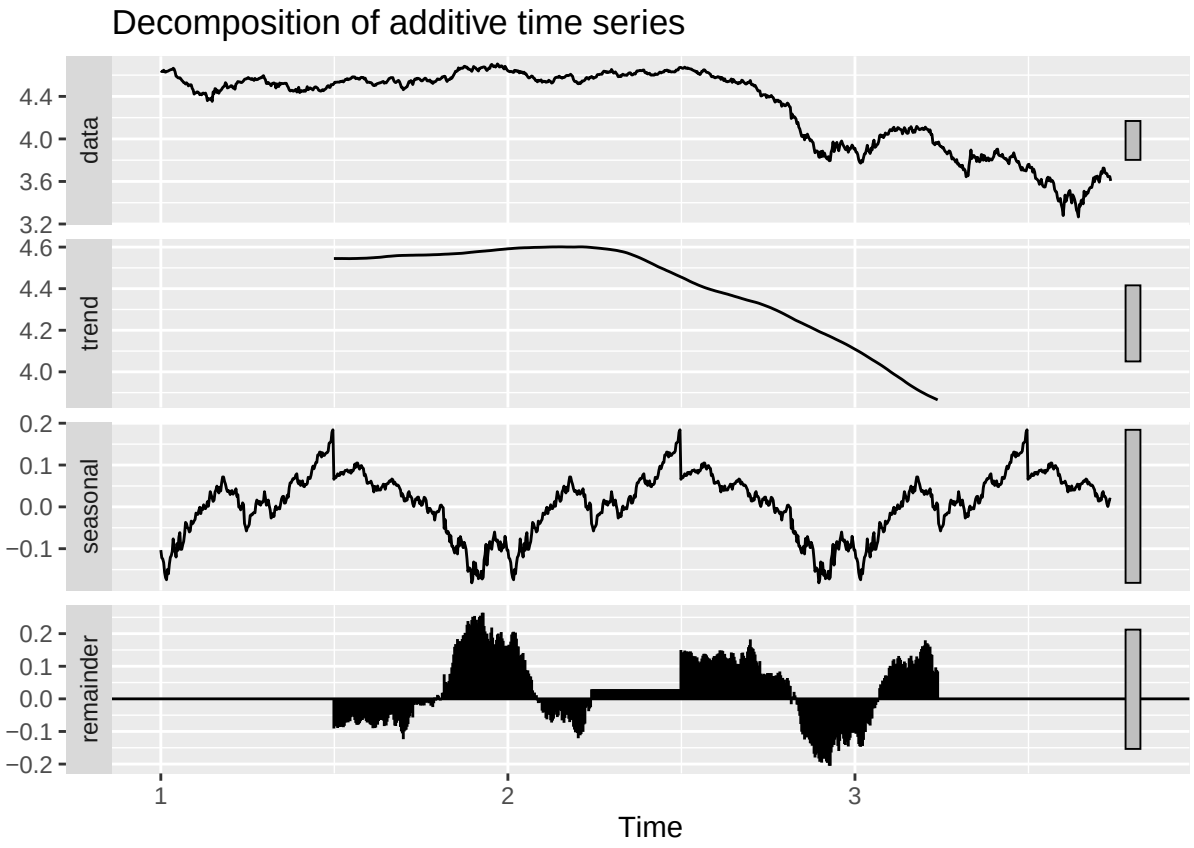
```
# Creating a new data frame called 'df' without the past contracts information.

# According to the number of columns and their names in 'oil_col_names', lets
# remove the columns 2 to 15 representing the 'CL#_log's variables
df = oil[,-2:-15]
head(df, 3)
```

```
##   OILPRICE BDIY_log  SPX_log  DX1_log  GC1_log  HO1_log USCI_log  GNR_log
## 1 4.640923 6.850126 7.221624 4.386554 7.413367 1.136197 4.108412 3.917806
## 2 4.633077 6.850126 7.235309 4.379762 7.419680 1.152564 4.120986 3.942552
## 3 4.634049 6.879356 7.222756 4.387449 7.418481 1.155182 4.115127 3.923952
##   SHCOMP_log FTSE_log  respLAG
## 1   7.744539 8.636699 4.631812
## 2   7.762536 8.650062 4.640923
## 3   7.766061 8.639729 4.633077
```

Plotting the `OILPRICE` time series considering the period of 365 days (1 year)

```
ts(df$OILPRICE, frequency = 365) %>%
  decompose() %>%
  autoplot()
```




---

Getting the number of rows (observations) and columns (variables) there are in the `df` data frame:

```
paste0("The 'df' data frame has ", nrow(df), " observations and  ", ncol(df),
       " variables. ")
```

```
## [1] "The 'df' data frame has 1000 observations and  11 variables. "
```

'df' data frame column names:

```
# columns names (df)
df_col_names <- colnames(df)
cat('Variables in the \'df\' data frame: \n\n')
```

```
## Variables in the 'df' data frame:
```

```
for (i in (1:ncol(df))) {print(paste0("Column number: ",i, ". Variable name: ",
                                      df_col_names[i]))}
```

```
## [1] "Column number: 1. Variable name: OILPRICE"
## [1] "Column number: 2. Variable name: BDIY_log"
## [1] "Column number: 3. Variable name: SPX_log"
## [1] "Column number: 4. Variable name: DX1_log"
## [1] "Column number: 5. Variable name: GC1_log"
## [1] "Column number: 6. Variable name: HO1_log"
## [1] "Column number: 7. Variable name: USCI_log"
## [1] "Column number: 8. Variable name: GNR_log"
## [1] "Column number: 9. Variable name: SHCOMP_log"
## [1] "Column number: 10. Variable name: FTSE_log"
## [1] "Column number: 11. Variable name: respLAG"
```

```
# paste0("Variable name: ", col_names)
```

Printing a statistic summary to overview the `df` features values:

```
# Let us see some basic statistics of our new 'df' data frame
summary(df)
```

```
##     OILPRICE        BDIY_log        SPX_log         DX1_log
##  Min.   :3.266   Min.   :5.670   Min.   :7.153   Min.   :4.369
##  1st Qu.:3.966   1st Qu.:6.596   1st Qu.:7.354   1st Qu.:4.391
##  Median :4.517   Median :6.806   Median :7.531   Median :4.417
##  Mean   :4.309   Mean   :6.787   Mean   :7.481   Mean   :4.459
##  3rd Qu.:4.580   3rd Qu.:7.011   3rd Qu.:7.611   3rd Qu.:4.557
##  Max.   :4.705   Max.   :7.757   Max.   :7.664   Max.   :4.613
##     GC1_log         HO1_log           USCI_log        GNR_log
##  Min.   :6.956   Min.   :-0.1442   Min.   :3.650   Min.   :3.317
##  1st Qu.:7.089   1st Qu.: 0.6220   1st Qu.:3.838   1st Qu.:3.787
##  Median :7.159   Median : 1.0547   Median :4.021   Median :3.868
##  Mean   :7.192   Mean   : 0.8600   Mean   :3.962   Mean   :3.818
##  3rd Qu.:7.345   3rd Qu.: 1.1013   3rd Qu.:4.070   3rd Qu.:3.920
##  Max.   :7.491   Max.   : 1.1877   Max.   :4.148   Max.   :3.985
##    SHCOMP_log       FTSE_log        respLAG
##  Min.   :7.576   Min.   :8.568   Min.   :3.266
##  1st Qu.:7.652   1st Qu.:8.716   1st Qu.:3.966
##  Median :7.734   Median :8.778   Median :4.517
##  Mean   :7.840   Mean   :8.760   Mean   :4.310
##  3rd Qu.:8.032   3rd Qu.:8.813   3rd Qu.:4.580
##  Max.   :8.550   Max.   :8.868   Max.   :4.705
```

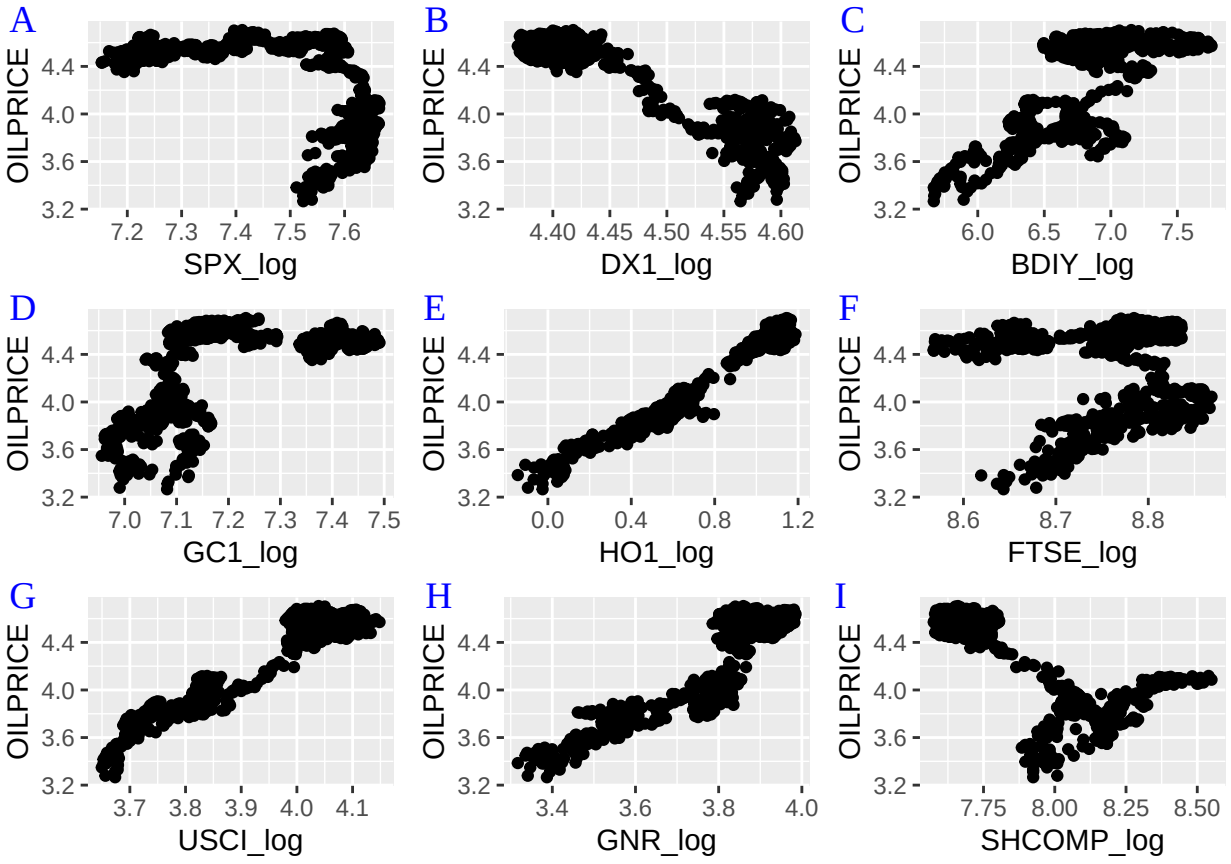
It seems we don't have any outliers or any missing inputs values in `df`. Let's continue our exploratory analysis.

**Plotting `OILPRICE` relation with other variables** To create a strategy that can solve our problem, we need to understand the relationship between Y and X's, the relation between the predictor variable and the response variables. For that will can use a scatter plot to make the relation graphically explicit:

```
p1 <- ggplot(df, aes(SPX_log, OILPRICE)) + geom_point()
p2 <- ggplot(df, aes(DX1_log, OILPRICE)) +  geom_point()
p3 <- ggplot(df, aes(BDIY_log, OILPRICE)) +  geom_point()
p4 <- ggplot(df, aes(GC1_log, OILPRICE)) +  geom_point()
p5 <- ggplot(df, aes(HO1_log, OILPRICE)) +  geom_point()
p6 <- ggplot(df, aes(FTSE_log, OILPRICE)) +  geom_point()
p7 <- ggplot(df, aes(USCI_log, OILPRICE)) +  geom_point()
p8 <- ggplot(df, aes(GNR_log, OILPRICE)) +  geom_point()
p9 <- ggplot(df, aes(SHCOMP_log, OILPRICE)) +  geom_point()

plot_list = list(p1, p2, p3, p4, p5, p6, p7, p8, p9)

plot_grid(plotlist = plot_list,
          labels = c('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I'),
          label_x = 0.01,
          label_y = 0.87,
          hjust = -0.5,
          vjust = -0.5,
          label_fontfamily = "serif",
          label_fontface = "plain",
          label_colour = "blue")
```




The scatter plot above shows different relations between the dependent (Y) and independent (X's) variables:

| Graph Label | Variable | Correlation |
|---|---|---|
| A | SPX_log | Unknown Correlation |
| B | DX1_log | Negative Linear Correlation |
| B | BDIY_log | Positive Linear Correlation |
| D | GC1_log | Unknown Correlation |
| E | HO1_log | Positive Linear Correlation |
| F | FTSE_log | Unknown Correlation |
| G | USCI_log | Positive Linear Correlation |
| H | GNR_log | Positive Linear Correlation |
| I | SHCOMP_log | Unknown Correlation |

In general, we expect to see a linear response from the variables 'DX1_log', 'HO1_log', 'USCI_log' and 'GNR_log' and maybe a non-linear correlation from 'SPX_log', 'GC1_log', 'FTSE_log' and 'SHCOMP_log'.

We'll have to find more about the variables to select the best treatment.

---

**Variables Distribution** To select the best suitable correlation test and to better interpret what the results mean, we have to know the variables' values distribution.

For this, we will plot a series of histograms to visualize their distribution and finally be able to select a proper correlation method to ensure good results when modeling.

```
# Making the objects in the 'df' data frame searchable
attach(df)

# Selecting plot's parameters
color1 <- 'black'
fill1 <- 'white'
alp <- .2
fill2 <- "#FF6666"
x_inter = mean(OILPRICE)
color2 <- 'blue'
lt <- 'dashed'
sz <- 1

# Plotting
par(mfrow=c(3,4))

h1 <- ggplot(df, aes(x=OILPRICE))+ ggtitle("Oil Price")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(OILPRICE)), color=color2, linetype=lt, size=sz)
h2 <- ggplot(df, aes(x=BDIY_log))+ ggtitle("Baltic Dry")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(BDIY_log)), color=color2, linetype=lt, size=sz)
h3 <- ggplot(df, aes(x=SPX_log))+ ggtitle("S&P 500 index")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(SPX_log)), color=color2, linetype=lt, size=sz)
```

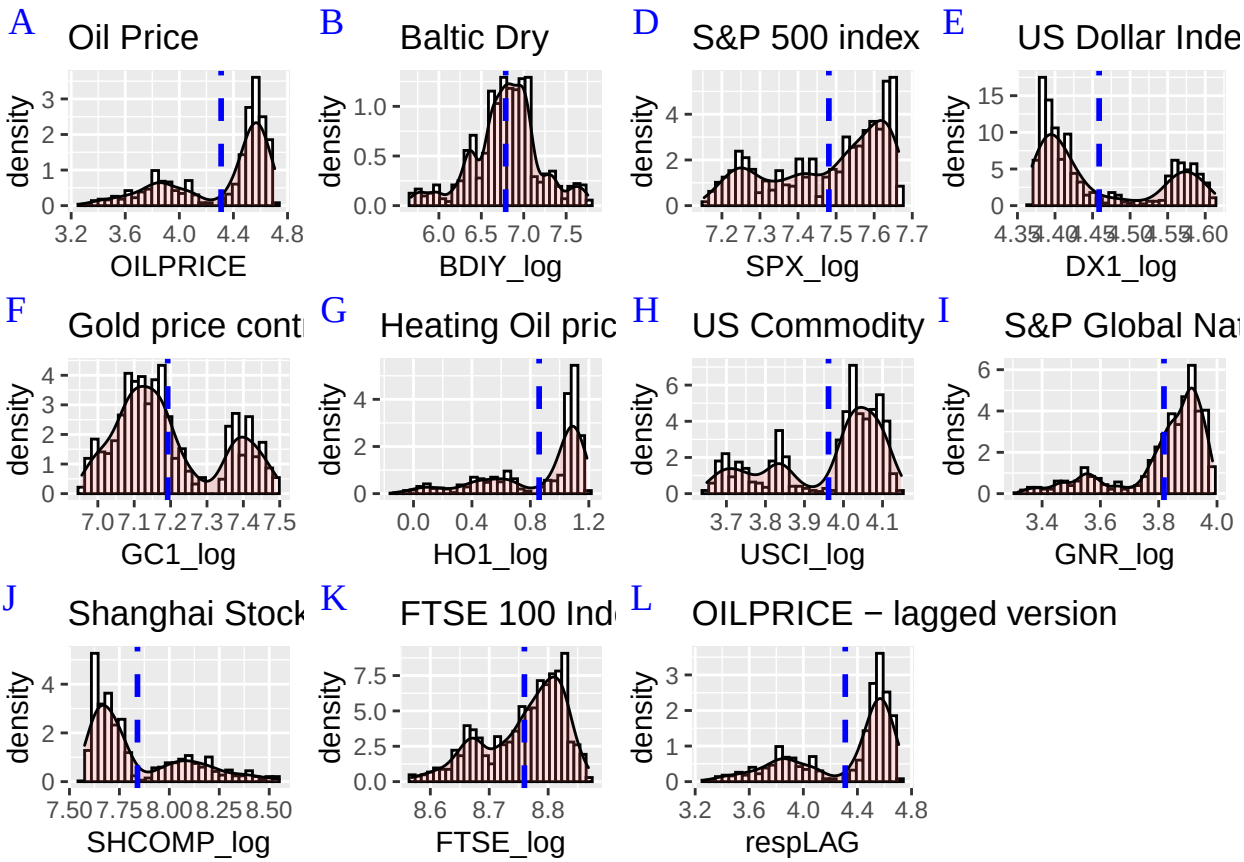
```
h4 <- ggplot(df, aes(x=DX1_log))+ ggtitle("US Dollar Index")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(DX1_log)), color=color2, linetype=lt, size=sz)
h5 <- ggplot(df, aes(x=GC1_log))+ ggtitle("Gold price contract trades")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(GC1_log)), color=color2, linetype=lt, size=sz)
h6 <- ggplot(df, aes(x=HO1_log))+ ggtitle("Heating Oil price contract traded")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(HO1_log)), color=color2, linetype=lt, size=sz)
h7 <- ggplot(df, aes(x=USCI_log))+ ggtitle("US Commodity Index")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(USCI_log)), color=color2, linetype=lt, size=sz)
h8 <- ggplot(df, aes(x=GNR_log))+ ggtitle("S&P Global Natural Resources Index")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(GNR_log)), color=color2, linetype=lt, size=sz)
h9 <- ggplot(df, aes(x=SHCOMP_log))+
  ggtitle("Shanghai Stock Exchange Composite Index")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(SHCOMP_log)), color=color2, linetype=lt,
             size=sz)
h10 <- ggplot(df, aes(x=FTSE_log))+ ggtitle("FTSE 100 Index")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(FTSE_log)), color=color2, linetype=lt, size=sz)
h11 <- ggplot(df, aes(x=respLAG))+ ggtitle("OILPRICE - lagged version")+
  geom_histogram(aes(y=..density..), colour=color1, fill=fill1)+
  geom_density(alpha=alp, fill=fill2)+
  geom_vline(aes(xintercept=mean(respLAG)), color=color2, linetype=lt, size=sz)

hist_list = list(h1, h2, h3, h4, h5, h6, h7, h8, h9, h10, h11)

plot_grid(plotlist = hist_list,
          labels = c('A', 'B', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M'),
          label_x = 0.01,
          label_y = 0.87,
          hjust = -0.5,
          vjust = -0.5,
          label_fontfamily = "serif",
          label_fontface = "plain",
          label_colour = "blue")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




With the exception of the 'BDIY_log', most of the distributions don't follow a Gaussian (normal) distribution.

---

**Variables Correlation** As seen, most of the variables don't follow a normal distribution, so it would be an assertive strategy to make the assumption that the variables' parameters are distribution-free. For that, we shall use a non-parametric approach to better suit the different distributions. Since we have quantitative non-normal variables on input and the same type in the output, the `Spearman test` would be a great pick for comparing these features.

The building of the correlation matrix using the Spearman test can help us understand the correlations and which features have significant levels, and which don't.

We'll use the P-value to interpret the results. The null hypothesis is that there is no relation whatsoever between the variables (correlation = 0).

P-test interpretation wise: higher p-values mean correlations closer to zero and low p-values mean correlation different than zero between the variables and Y.

```r
# Building a correlation Matrix
cor.mtest <- function(mat, ...) {
    mat <- as.matrix(mat)
    n <- ncol(mat)
    p.mat<- matrix(NA, n, n)
    diag(p.mat) <- 0
    for (i in 1:(n - 1)) {
        for (j in (i + 1):n) {
            tmp <- cor.test(mat[, i], mat[, j], ...)
            p.mat[i, j] <- p.mat[j, i] <- tmp$p.value
        }
    }
  colnames(p.mat) <- rownames(p.mat) <- colnames(mat)
  p.mat
}

# matrix of the p-value of the correlation
p.mat <- cor.mtest(df, method = 'spearman', exact=FALSE)
p.mat
```

```
##                OILPRICE     BDIY_log       SPX_log       DX1_log       GC1_log
## OILPRICE    0.000000e+00 9.878893e-94  1.164449e-62 3.393567e-222  4.534907e-90
## BDIY_log    9.878893e-94 0.000000e+00  6.792721e-26  1.025300e-66  3.120209e-12
## SPX_log     1.164449e-62 6.792721e-26  0.000000e+00 4.618178e-122 5.540047e-294
## DX1_log    3.393567e-222 1.025300e-66 4.618178e-122  0.000000e+00 7.575202e-189
## GC1_log     4.534907e-90 3.120209e-12 5.540047e-294 7.575202e-189  0.000000e+00
## HO1_log    1.329004e-258 2.946376e-60 1.383726e-143 2.347075e-294 3.759788e-200
## USCI_log   9.042281e-178 6.261781e-36 1.183454e-141 5.987330e-298 5.283737e-256
## GNR_log    1.652549e-190 5.182126e-32  8.890018e-61 1.727438e-279 1.763336e-151
## SHCOMP_log 4.635703e-193 6.546882e-72 1.049770e-111 6.663341e-202 3.874057e-110
## FTSE_log    3.288032e-08 2.359658e-03 8.597015e-120  1.514743e-03  9.516503e-38
## respLAG     0.000000e+00 1.454298e-94  6.127019e-62 5.106686e-225  3.118096e-90
##                 HO1_log      USCI_log       GNR_log    SHCOMP_log
## OILPRICE   1.329004e-258 9.042281e-178 1.652549e-190 4.635703e-193
## BDIY_log    2.946376e-60  6.261781e-36  5.182126e-32  6.546882e-72
## SPX_log    1.383726e-143 1.183454e-141  8.890018e-61 1.049770e-111
## DX1_log    2.347075e-294 5.987330e-298 1.727438e-279 6.663341e-202
## GC1_log    3.759788e-200 5.283737e-256 1.763336e-151 3.874057e-110
## HO1_log     0.000000e+00 1.512872e-270 3.812661e-208 2.106332e-169
## USCI_log   1.512872e-270  0.000000e+00 1.232274e-298 3.848899e-163
## GNR_log    3.812661e-208 1.232274e-298  0.000000e+00 3.509312e-120
## SHCOMP_log 2.106332e-169 3.848899e-163 3.509312e-120  0.000000e+00
## FTSE_log    2.122789e-03  4.528966e-07  7.927264e-06  2.344613e-03
## respLAG    1.779613e-263 5.687291e-180 2.645174e-194 1.610871e-189
##                 FTSE_log       respLAG
## OILPRICE    3.288032e-08  0.000000e+00
## BDIY_log    2.359658e-03  1.454298e-94
## SPX_log    8.597015e-120  6.127019e-62
## DX1_log     1.514743e-03 5.106686e-225
## GC1_log     9.516503e-38  3.118096e-90
## HO1_log     2.122789e-03 1.779613e-263
## USCI_log    4.528966e-07 5.687291e-180
## GNR_log     7.927264e-06 2.645174e-194
```

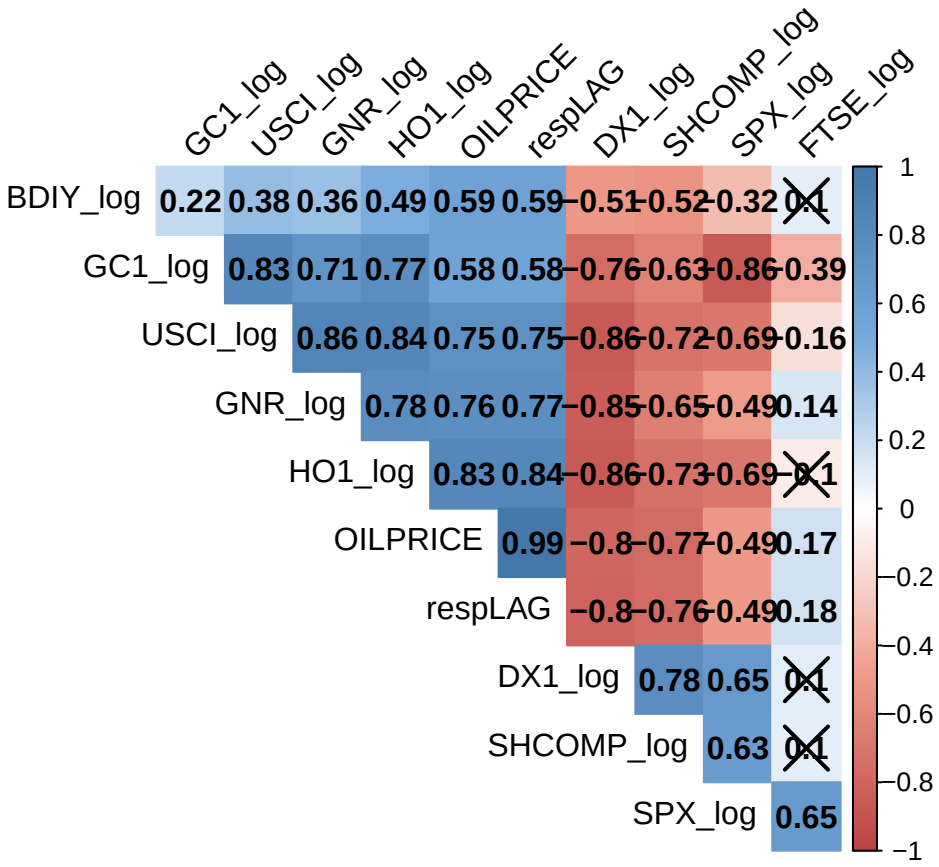
```
## SHCOMP_log  2.344613e-03 1.610871e-189
## FTSE_log    0.000000e+00  2.247735e-08
## respLAG     2.247735e-08  0.000000e+00
```

As we can see from the correlation matrix above, almost all of the features have extremely low values. That means that they have a correlation different than zero to the Y variable. But when we take a look at the `FTSE_log` feature, we notice some relatively higher values compared to the other features.

If we select a threshold (0,1%) to the significance levels (the probability of rejecting the null hypothesis given that it is true), we will notice that some of the FTSE 100 Indexes stay above of the threshold. This can be seen below in the graphical correlation matrix. The squares marked with an 'X' mean that the value is higher than the threshold. In addition, all the other `FTSE_log` correlation values are kinda low compared to other features.

We can say that all but the FTSE 100 index have a different than zero correlation with `OILPRICE` variable.

```
corr_mat=cor(df,method="s")
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
corrplot(corr_mat, method="color", col=col(200),
         type="upper", order="hclust",
         addCoef.col = "black",
         tl.col="black", tl.srt=45,
         # Combine with significance
         p.mat = p.mat, sig.level = 0.001, insig = "pch",
         # hide correlation coefficient on the principal diagonal
         diag=FALSE
         )
```



## 5 - DATA PREPARATION

**Variables Transformation** As informed in the dataset description, the data values are already logged transformations of the original format. This means that we don't need to do any other treatment to the data and we can model it as it is. In addition, any other additional transformation would difficult the model interpretation and unnecessarily increase its complexity.

**GAMLSS Model** The GAMLSS library is well suited for our data because of its ability to handle asymmetric distributions, including high skewed and/or kurtotic distributions. The package is also ready to deal with addictive non-parametric functions.

For this project, we'll use the RS algorithm. It usually works with four parameters: location (`mu`), scale (`sigma`), and two shape parameters (`nu` and `tau`), which are estimated by a penalizing likelihood function `k`. The model also accepts additive terms such as smoothing functions for the continuous explanatory variable x

**Family used and performance indicators** For the model to work properly we need to evaluate which distribution will be passed to it. This was a process of trial and error between the long list of different distributions available in the GAMLSS library. The complete list of the distribution is linked in the Bibliography section.

A lot of different distributions were tested and the Box-Cox power exponential (BCPE) presented some of the lowest non-parametric test values (Global Deviation - GD, Akaike Information Criterion - AIC, and Schwarz Bayesian Criterion - SBC) among all while well-fitting the OILPRICE distribution curve. Briefly, Box-Cox transformations tries to transform our data into a normal distribution or at least close to a normal distribution.

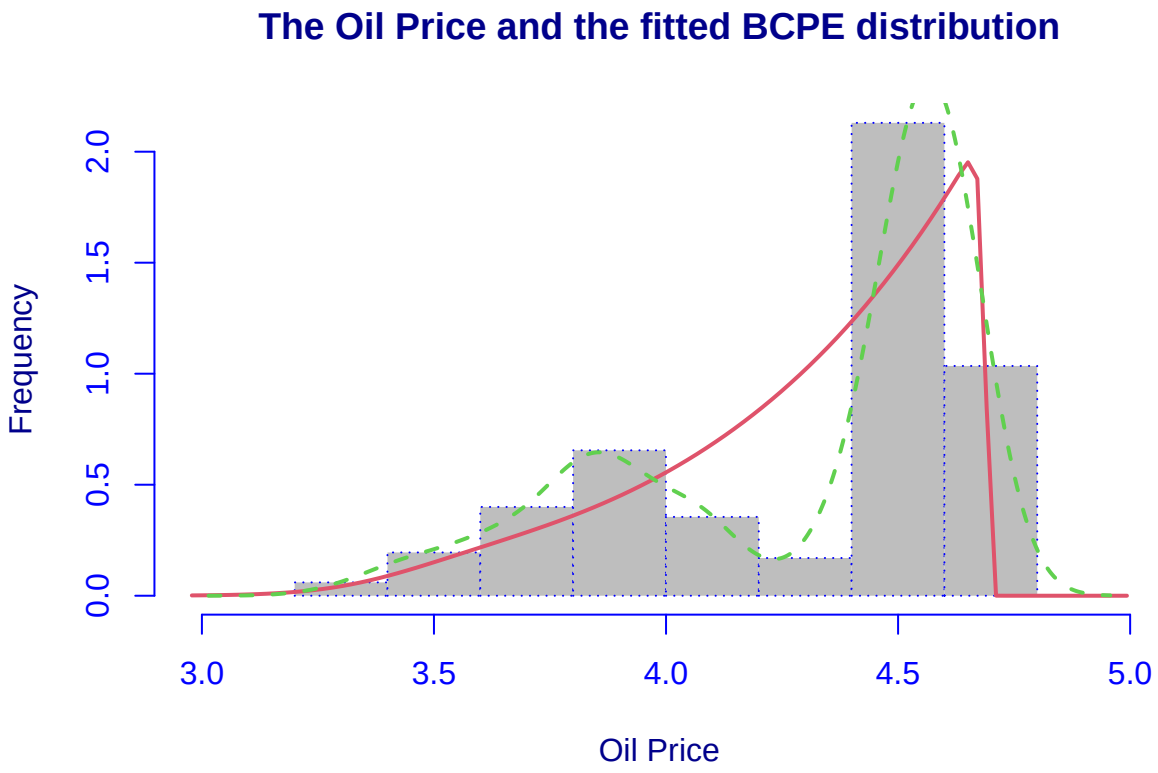
Functions used with the BCPE family distribution:

```
# showing available link functions for BCPE distribution parameters
show.link('BCPE')
```

```
## $mu
## c("inverse", "log", "identity", "own")
##
## $sigma
## c("inverse", "log", "identity", "own")
##
## $nu
## c("inverse", "log", "identity", "own")
##
## $tau
## c("logshiftto1", "log", "identity", "own")
```

In the graph below, the red line represents the Box-Cox Power Exponential (BCPE) compared with the `OILPRICE` distribution (green line). They show some resemblance and seem sufficiently good for modeling.

```
histDist(df$OILPRICE,
         family = BCPE,
         density = TRUE,
         main = 'The Oil Price and the fitted BCPE distribution',
         ylab = 'Frequency',
         xlab = 'Oil Price')
```




```
##
## Family:  c("BCPE", "Box-Cox Power Exponential")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = df$OILPRICE, family = "BCPE")
##
## Mu Coefficients:
## [1]  4.374
## Sigma Coefficients:
## [1]  -2.877
## Nu Coefficients:
## [1]  9.382
## Tau Coefficients:
## [1]  3.339
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   996
## Global Deviance:     65.6076
##             AIC:     73.6076
```

```
##              SBC:     93.2387
```

---

### 6 - MODEL TRAINING

While the construction of the project I came to know that the past traded contracts ends up inserting certain amount of noise into the data, negatively influencing the final model's performance. To avoid this issue, to keep the simplicity of the project, and to achieve better computational efficiency and aiming to solve the proposed problem, we'll have only one model based on the `df` data frame. I will not use any additive terms either, the models without them had slightly better results. Also, keeping only one model allows us to add more algorithm training cycles to achieve better results while maintaining a relatively low script execution time.

- `model_df`: used the `df` data frame and removed the past traded contracts and leaving only the lagged variable with no additive terms applied.
- Train/test split: 70% train, 30% test

```
# Dividing the 'df' data frame into train and test
dt_df = sample(nrow(df), nrow(df)*.7, replace = FALSE)
train_df <- df[dt_df, ]
test_df <- df[-dt_df, ]

print('Data splitted: 70% train, 30% test')
```

```
## [1] "Data splitted: 70% train, 30% test"
```

Definition of the model using BCPE as the family of distribution and 350 algorithm cycles. In my case, the algorithm converged at approximately 240 cycles.

I built a conditional check to load the trained model if it exists or to refit if it doesn't.

```
## check if model exists? If not, refit:
if(file.exists("gamlss_oil_model.rda")) {
    ## load model
    load("gamlss_oil_model.rda")
} else {
    ## (re)fit the model
    model_df <- gamlss(formula = OILPRICE ~ ., family = BCPE, data = train_df,
                       control = gamlss.control(n.cyc = 350))
}

# save the model
save(model_df, file = "gamlss_oil_model.rda")
```

Printing a statistic summary on the model

```
summary(model_df)
```

```
## ******************************************************************
## Family:  c("BCPE", "Box-Cox Power Exponential")
##
## Call:  gamlss(formula = OILPRICE ~ ., family = BCPE, data = train_df,
##     control = gamlss.control(n.cyc = 350))
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.447840   0.222206   2.015 0.044248 *
## BDIY_log    -0.001767   0.001683  -1.050 0.294105
## SPX_log     -0.101904   0.012123  -8.406 2.43e-16 ***
## DX1_log     -0.021347   0.022834  -0.935 0.350173
## GC1_log     -0.068064   0.011230  -6.061 2.23e-09 ***
## HO1_log      0.004727   0.010092   0.468 0.639656
## USCI_log     0.063255   0.017820   3.550 0.000412 ***
## GNR_log      0.009219   0.017327   0.532 0.594862
## SHCOMP_log  -0.003182   0.004702  -0.677 0.498855
## FTSE_log     0.093224   0.017879   5.214 2.44e-07 ***
## respLAG      0.960363   0.007368 130.339  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.14670    0.04404  -116.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Nu link function:  identity
## Nu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -4.405      7.251  -0.607    0.544
##
## ------------------------------------------------------------------
## Tau link function:  log
## Tau Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.30652    0.05447  -5.627 2.66e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  700
## Degrees of Freedom for the fit:  14
##       Residual Deg. of Freedom:  686
##                       at cycle:  240
##
```

```
## Global Deviance:     -3434.44
##              AIC:     -3406.44
##              SBC:     -3342.725
## ****************************************************************
```

Due to the elevated significance values presented at the correlation matrix and its respective plot, we have to be aware of signs of overfitting. Therefore, the models' evaluation metrics AIC and SBC are not the best metrics to evaluate the model's performance. Global deviance is a good performance metric, but again, it also can suffer the effects of possible overfitting.

---

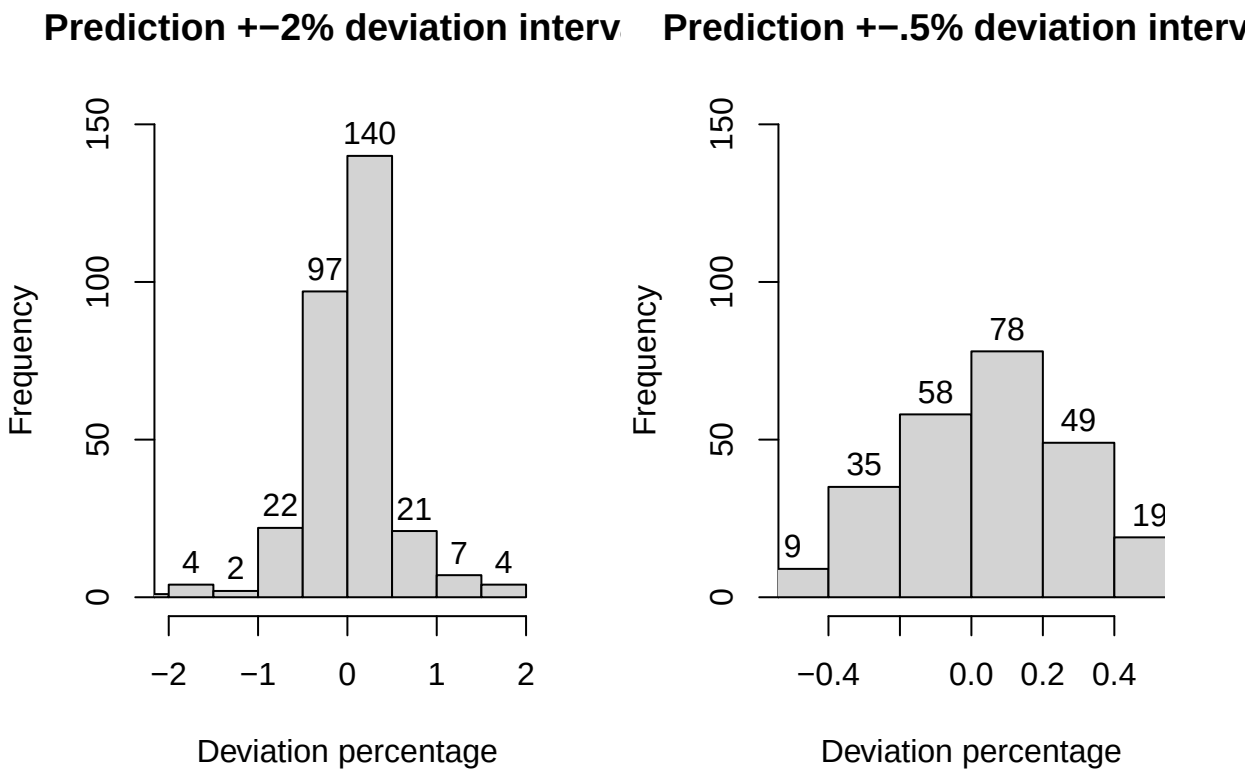
## 7 - MODEL EVALUATION

Now let's predict the test subset, calculate its percentage deviation and analyze the results.

```
# Prediction
test_df$pred <- predict(model_df, newdata=test_df, type = "response")

# Deviation calculation
test_df$pred.deviation  <- round(test_df$pred/test_df$OILPRICE,10)
test_df$pred.deviation  <- test_df$pred.deviation-1

# Plotting prediction's deviation
par(mfrow=c(1,2))
hist(100*test_df$pred.deviation, main="Prediction +-2% deviation interval",
     xlab = "Deviation percentage", labels = T,
     xlim =c(-2,2),
     ylim = c(0,150))
hist(100*test_df$pred.deviation, main="Prediction +-.5% deviation interval",
     xlab = "Deviation percentage", labels = T,
     xlim =c(-0.5,0.5),
     ylim = c(0,150), breaks = 32)
```




The interval of 1% deviation of the predictions shows us an important metric to evaluate the model. We can see that the range of deviation in the first graph is 2%. Considering that we are dealing with price values, a 2% deviation is a very high deviation. Given the volume of negotiations and the values involved, the expected oscillation in this business segment is about 1%. An excellent deviation oscillation range would be considered about 0.5%.

Also, this information is a good metric to evaluate the model's performance. The ideal scenario would be to feed additional information to the model to see if it is overfitted. However, while elaborating on this project, I came across different models and different models metrics and I'm confident that the model is not under fitted or overfitted.

Finally, let's see the percentage of values between -+1% and +-0.5% deviation range and analyze if the results are sufficient for our problem.

```
test_dev <- 100*test_df$pred.deviation
test_1dev <- sum(abs(test_dev) <= 1)
test_05dev <- sum(abs(test_dev) <= 0.5)
test_size <- nrow(test_df)

test_in_1dev = (test_1dev / test_size)*100
test_in_05dev = (test_05dev / test_size)*100

paste0(round(test_in_1dev,2),
       '% of the predicted values are in a +-1% deviation margin to the real values')
```

```
## [1] "93.33% of the predicted values are in a +-1% deviation margin to the real values"
```

```
paste0(round(test_in_05dev,2),
       '% of the predicted values are in a +-0.5% deviation margin to the real values')
```

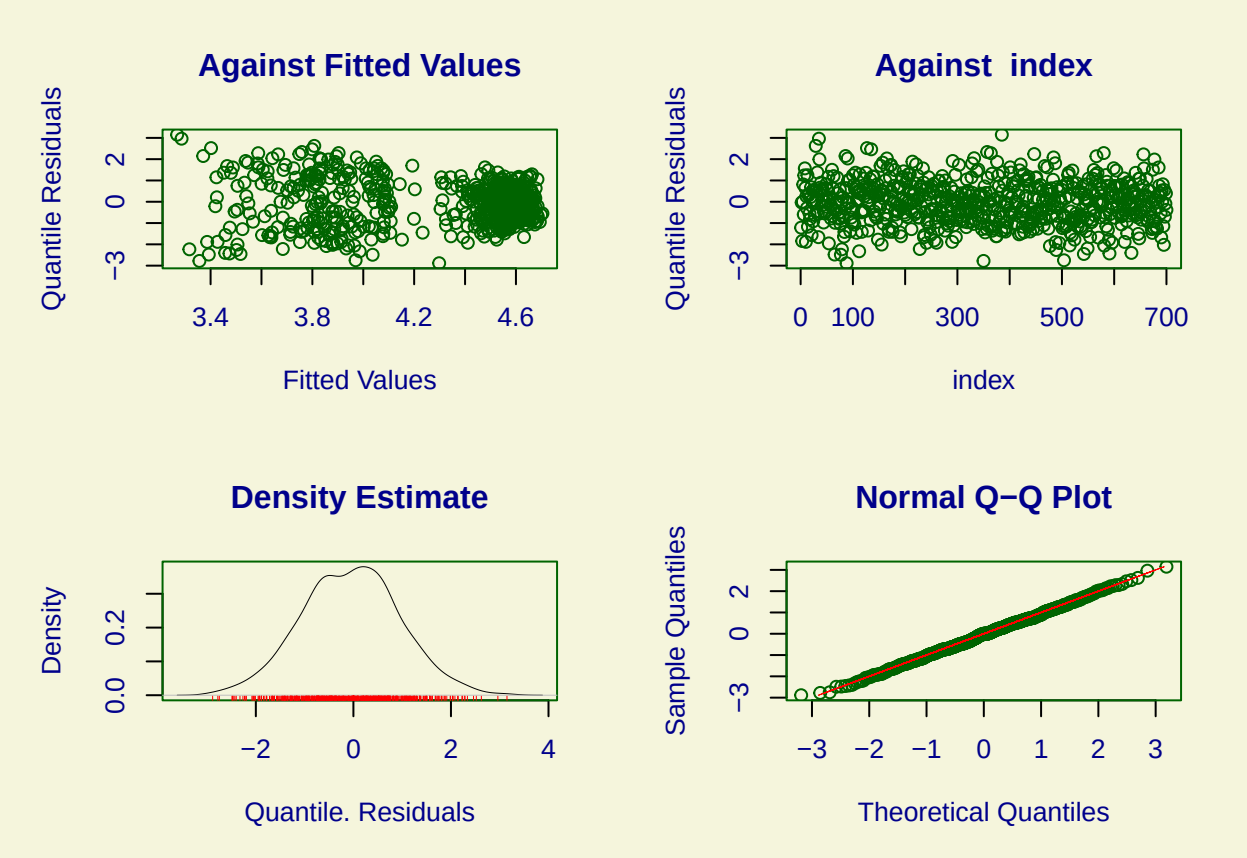
```
## [1] "79% of the predicted values are in a +-0.5% deviation margin to the real values"
```

We have achieved our goal! 93% of the predicted values were +-1% in a deviation margin, and 79% in a +-0.5% margin.

---

**Residuals** Errors and residuals are two closely related and easily confused measures of the deviation. The error is the predicted value minus the true value and the residual is the difference between the observed value and the estimated value of the quantity of interest

```
plot(model_df)
```



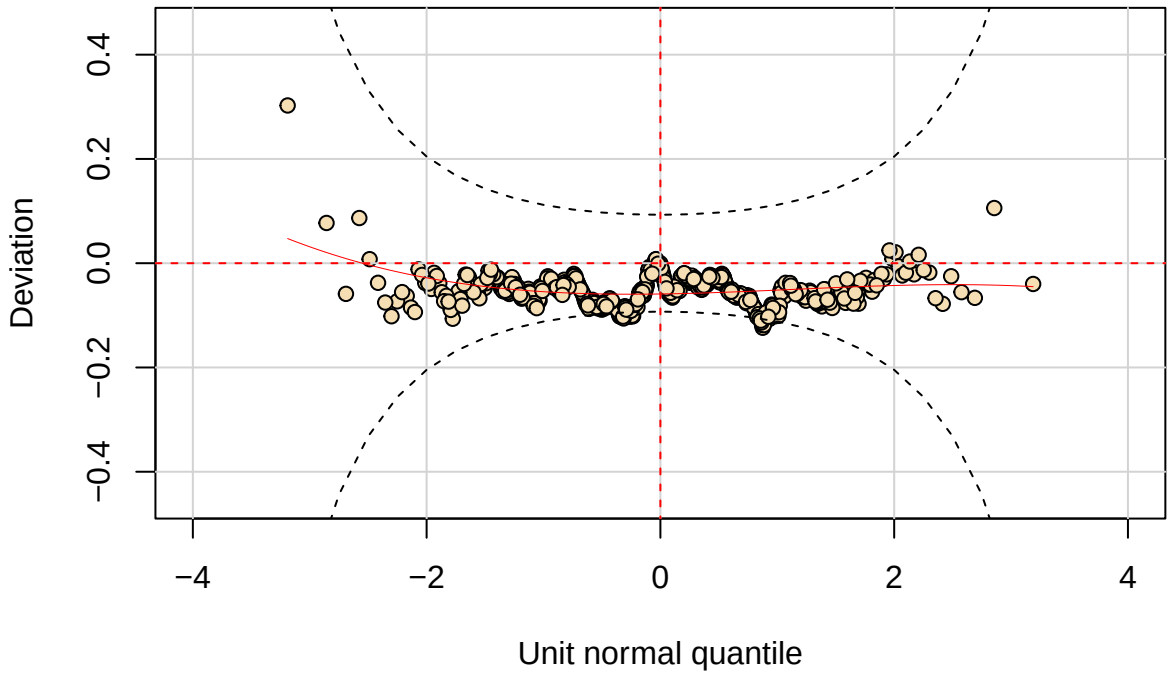

```
## ************************************************************************
##         Summary of the Quantile Residuals
##                              mean   =  -0.05313394
##                          variance   =  0.9967813
##                 coef. of skewness   =  0.03337896
##                 coef. of kurtosis   =  2.92972
## Filliben correlation coefficient    =  0.9995241
## ************************************************************************
```

As we can see, the density estimate follows a normal distribution, and consequently, the observed sample percentiles of the residuals are linear. This means that the error terms are normally distributed.

---

**Worm Plot** The same thing reflects on the worm plot, where all of the residuals are in or at the border of the dotted confidence bands. This indicates the adequacy in modeling the distribution.

```
# Plotting residuals
wp(model_df)
```




---

**Metrics: Non-parametric Tests** Let's see the non-parametric numbers

```
# Getting each model's non parametric test

GD = c(model_df$G.deviance) #Deviation
AIC = c(model_df$aic)# Overfitting
SBC = c(model_df$sbc)# Overfitting

paste0('The model Global Deviation - GD was: ', round(GD,2))
```

```
## [1] "The model Global Deviation - GD was: -3434.44"
```

```
paste0('The model Akaike Information Criterion - AIC was: ', round(AIC,2))
```

```
## [1] "The model Akaike Information Criterion - AIC was: -3406.44"
```

```
paste0('The model Schwarz Bayesian Criterion - SBC was: ', round(SBC,2))
```

```
## [1] "The model Schwarz Bayesian Criterion - SBC was: -3342.73"
```

---

Building a daily_oil_prediction.csv file with the original and predicted data.

```
write.csv(test_df,"daily_oil_prediction.csv", row.names = FALSE)
```

---

## 9 - CONCLUSIONS

Not using the past contracts variables `CL#_log` was a good strategy to not impose unnecessary noise into the data. The features used in `df` data frame were sufficiently good and at the same time kept the model simple and effective. The high significance in the correlation matrix imposed a doubt whether the model had overfitted or not but the model results on the test data and the robust analysis on its residuals supports the idea that the model is just on the right spot. The model returned a high performance on predicting the values, delivering a high-quality standard prevision, and more important: answered the problem question and achieved the goals.

As said during the execution of this project, the model could predict 93% of the values within 1% or less deviation margin, and 78% accuracy on 0.5% deviation margin. Maybe some tuning on the weights and control parameters of the model could have improved the results, and maybe some additional tuning on the previous models that I used with additive terms such as Cubic Splines could deliver better results. But as the time was a little bit short and I'm continuously learning, maybe I'll come back soon to this project to continue improving its results.

## 10 - PERSONAL CONCLUSION

It was a very challenging project and I'm very glad that I could achieve good results. I tried to bring some of the research methods that I find relevant to the real job market while keeping it simple so that you, the reader, could have a enjoyable experience.

I hope you liked it! Any feedback doesn't hesitate to contact me on my LinkedIn or check out my other projects at my github page:

LinkedIn profile: https://www.linkedin.com/in/ivan-berlim-gonçalves/?locale=en_US

Github: https://github.com/ivanbergon

## 11 - BIBLIOGRAPHY


- Stasinopoulos, M., Rigby, R. A., & Akantziliotou, C. (2008). Instructions on how to use the GAMLSS package in R. 206. http://www.gamlss.com/wp-content/uploads/2013/01/gamlss-manual.pdf

- Rigby, R., Stasinopoulos, M., Heller, G., & De Bastiani, F. (2019). Distributions for Modelling Location, Scale and Shape: Using GAMLSS in R. CRC Press. http://www.gamlss.com/wp-content/uploads/2018/01/DistributionsForModellingLocationScaleandShape.pdf
- Stasinopolulos, M., Rigby, R. A., Voudouris, V., Heler, G., & Bastiani De, F. (2017). Flexible regression and smoothing the GAMLSS packages in R. 571. http://www.gamlss.com/wp-content/uploads/2015/07/FlexibleRegressionAndSmoothingDraft-1.pdf
- Scandroglio, Giacomo & Gori, Andrea & Vaccaro, Emiliano & Voudouris, Vlasios. (2013). Estimating VaR and ES of the spot price of oil using futures-varying centiles. Int. J. of Financial Engineering and Risk Management.

1. 6 - 19. 10.1504/IJFERM.2013.053713. https://www.researchgate.net/publication/264815334_Estimating_VaR_and_ES_of_the_spot_price_of_oil_using_futures-varying_centiles

- RDRR.io. oil: The oil price data. https://rdrr.io/cran/gamlss.data/man/oil.html
- RDocumentation. gamlss.family: Family Objects for fitting a GAMLSS model. https://www.rdocumentation.org/packages/gamlss.dist/versions/5.3-2/topics/gamlss.family
- Penn State - Eberly College of Science. Normal Probability Plot of Residuals. https://online.stat.psu.edu/stat462/node/122/