oil-data-analysis

niranjana

2022-04-26

```
## Loading required package: splines
```

```
## Loading required package: gamlss.data
```

```
##
## Attaching package: 'gamlss.data'
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
## Loading required package: gamlss.dist
```

```
## Loading required package: MASS
```

```
## Loading required package: nlme
```

```
## Loading required package: parallel
```

```
##  **********   GAMLSS Version 5.4-1  **********
```

```
## For more on GAMLSS look at https://www.gamlss.com/
```

```
## Type gamlssNews() to see new features/changes/bug fixes.
```

```
## Loading required package: mgcv
```

```
## This is mgcv 1.8-39. For overview type 'help("mgcv-package")'.
```

```
## Loading required package: nnet
```

```
##
## Attaching package: 'nnet'
```

```
## The following object is masked from 'package:mgcv':
##
##     multinom
```

```
## Loading required package: rpart
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x dplyr::select()   masks MASS::select()
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
## The following object is masked from 'package:gamlss.add':
##
##     tr
```

```
## The following object is masked from 'package:gamlss':
##
##     cs
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
##
## Attaching package: 'cowplot'
```

```
## The following object is masked from 'package:lubridate':
##
##     stamp
```

```
##
## Attaching package: 'ggpubr'
```

```
## The following object is masked from 'package:cowplot':
##
##     get_legend
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:ggpubr':
##
##     gghistogram
```

```
## The following object is masked from 'package:gamlss':
##
##     CV
```

```
## The following object is masked from 'package:nlme':
##
##     getResponse
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following object is masked from 'package:psych':
##
##     describe
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

The Oil Data Set

Reference : https://rdrr.io/cran/gamlss.data/man/oil.html (https://rdrr.io/cran/gamlss.data/man/oil.html)

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 price 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.

```
#extract the oil dataset
data(oil)
```

A data frame with 1000 observations on the following 25 variables.

```
# Print Different Variable names in the data set
colnames(oil)
```

```
##  [1] "OILPRICE"   "CL2_log"    "CL3_log"    "CL4_log"    "CL5_log"
##  [6] "CL6_log"    "CL7_log"    "CL8_log"    "CL9_log"    "CL10_log"
## [11] "CL11_log"   "CL12_log"   "CL13_log"   "CL14_log"   "CL15_log"
## [16] "BDIY_log"   "SPX_log"    "DX1_log"    "GC1_log"    "HO1_log"
## [21] "USCI_log"   "GNR_log"    "SHCOMP_log" "FTSE_log"   "respLAG"
```

Description of Variables

`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.

`CL2_log`, `CL3_log`, `CL4_log`, `CL5_log`, `CL6_log`, `CL7_logCL8_log`, `CL9_log`, `CL10_log`, `CL11_log`, `CL12_log`, `CL13_log`, `CL14_log`, `CL15_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`: he 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.

Quickly view the database:

```
# Display first 5 rows of oil data
head(oil, n = 5)
```

```
##   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
## 4 4.646312 4.638315 4.642562 4.646120 4.648708 4.649952 4.650621 4.651099
## 5 4.631520 4.650526 4.654722 4.658047 4.660321 4.661267 4.661740 4.662117
##    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
## 4 4.651577 4.651004 4.649570 4.647080 4.644102 4.640923 4.636960 6.882437
## 5 4.662401 4.661645 4.659942 4.656908 4.653484 4.649761 4.645544 6.896694
##    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
## 4 7.222252 4.383675 7.410347 1.136743 4.103965 3.925531   7.765158 8.642292
## 5 7.237620 4.382364 7.399704 1.139946 4.107096 3.941970   7.755763 8.659907
##    respLAG
## 1 4.631812
## 2 4.640923
## 3 4.633077
## 4 4.634049
## 5 4.646312
```

There are no null values as observed below

```
# Check for null values
sum(is.na(oil))
```

```
## [1] 0
```

A quick summary of the database shows the mean of the response variable as 4.3089067 and its standard deviation is 0.37145840

```
# Show Summary Statistics for each of the variables in the dataset
skim(oil)
```

Data summary

Name	oil
Number of rows	1000
Number of columns	25
_____	
Column type frequency:	
numeric	25
_____	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
OILPRICE	0	1	4.31	0.37	3.27	3.97	4.52	4.58	4.71	▁▁▂▇
CL2_log	0	1	4.32	0.36	3.34	3.98	4.52	4.58	4.70	▁▁▂▇
CL3_log	0	1	4.32	0.35	3.39	4.00	4.52	4.58	4.68	▁▁▂▇
CL4_log	0	1	4.33	0.34	3.43	4.02	4.52	4.58	4.67	▁▁▂▇
CL5_log	0	1	4.33	0.33	3.48	4.03	4.52	4.57	4.67	▁▁▂▇
CL6_log	0	1	4.33	0.32	3.50	4.05	4.52	4.57	4.67	▁▁▂▇
CL7_log	0	1	4.33	0.31	3.52	4.06	4.52	4.56	4.67	▁▁▂▇
CL8_log	0	1	4.33	0.31	3.53	4.07	4.51	4.56	4.67	▁▁▂▇
CL9_log	0	1	4.33	0.30	3.54	4.08	4.51	4.55	4.67	▁▁▂▇
CL10_log	0	1	4.34	0.29	3.56	4.08	4.51	4.55	4.67	▁▁▂▇
CL11_log	0	1	4.33	0.29	3.57	4.09	4.51	4.54	4.67	▁▁▂▇

CL12_log	0	1	4.33	0.28	3.58	4.10	4.50	4.54	4.66
CL13_log	0	1	4.33	0.28	3.59	4.10	4.50	4.53	4.66
CL14_log	0	1	4.33	0.27	3.59	4.11	4.50	4.53	4.65
CL15_log	0	1	4.33	0.27	3.60	4.12	4.49	4.52	4.65
BDIY_log	0	1	6.79	0.40	5.67	6.60	6.81	7.01	7.76
SPX_log	0	1	7.48	0.15	7.15	7.35	7.53	7.61	7.66
DX1_log	0	1	4.46	0.08	4.37	4.39	4.42	4.56	4.61
GC1_log	0	1	7.19	0.14	6.96	7.09	7.16	7.35	7.49
HO1_log	0	1	0.86	0.33	-0.14	0.62	1.05	1.10	1.19
USCI_log	0	1	3.96	0.14	3.65	3.84	4.02	4.07	4.15
GNR_log	0	1	3.82	0.15	3.32	3.79	3.87	3.92	3.98
SHCOMP_log	0	1	7.84	0.24	7.58	7.65	7.73	8.03	8.55
FTSE_log	0	1	8.76	0.07	8.57	8.72	8.78	8.81	8.87
respLAG	0	1	4.31	0.37	3.27	3.97	4.52	4.58	4.71

```
# Since the data is daily data, let us add a index (0 -999) to the data
oil <- tibble::rowid_to_column(oil, "index")
# Display data with the Index
head(oil,5)
```

```
##   index OILPRICE  CL2_log  CL3_log  CL4_log  CL5_log  CL6_log  CL7_log  CL8_log
## 1     1 4.640923 4.636475 4.641116 4.644968 4.648038 4.649761 4.650908 4.651863
## 2     2 4.633077 4.645352 4.649857 4.653484 4.656338 4.657858 4.658711 4.659564
## 3     3 4.634049 4.637831 4.642466 4.646312 4.649665 4.651672 4.653103 4.654341
## 4     4 4.646312 4.638315 4.642562 4.646120 4.648708 4.649952 4.650621 4.651099
## 5     5 4.631520 4.650526 4.654722 4.658047 4.660321 4.661267 4.661740 4.662117
##    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
## 4 4.651577 4.651004 4.649570 4.647080 4.644102 4.640923 4.636960 6.882437
## 5 4.662401 4.661645 4.659942 4.656908 4.653484 4.649761 4.645544 6.896694
##    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
## 4 7.222252 4.383675 7.410347 1.136743 4.103965 3.925531   7.765158 8.642292
## 5 7.237620 4.382364 7.399704 1.139946 4.107096 3.941970   7.755763 8.659907
##    respLAG
## 1 4.631812
## 2 4.640923
## 3 4.633077
## 4 4.634049
## 5 4.646312
```

Methodology of the Analysis

There are two overarching steps used for EDA - visual analysis followed by statistical confirmations for visual observations.

The general steps include:

- Step 1 : Data Exploration
- Step 2 : Feature Engineering
- Step 3 : Feature Selection
- Step 4 : Model Testing and Validation

Step 1 : Exploratory Data Analysis

Part A : Visual Analysis

1. Explore the response variable

Steps:

- Show the variation of response variable with time
- Distribution of the response variable

2. Explore the covariates

- The trends of the covariates with time
- The distribution of the various response variables
- Trends of the covariates with the response variable

2. Anomalies in Oil Prices

- Anomalies in monthly Data
- Anomalies in Yearly Data

Part B : Statistical Analysis

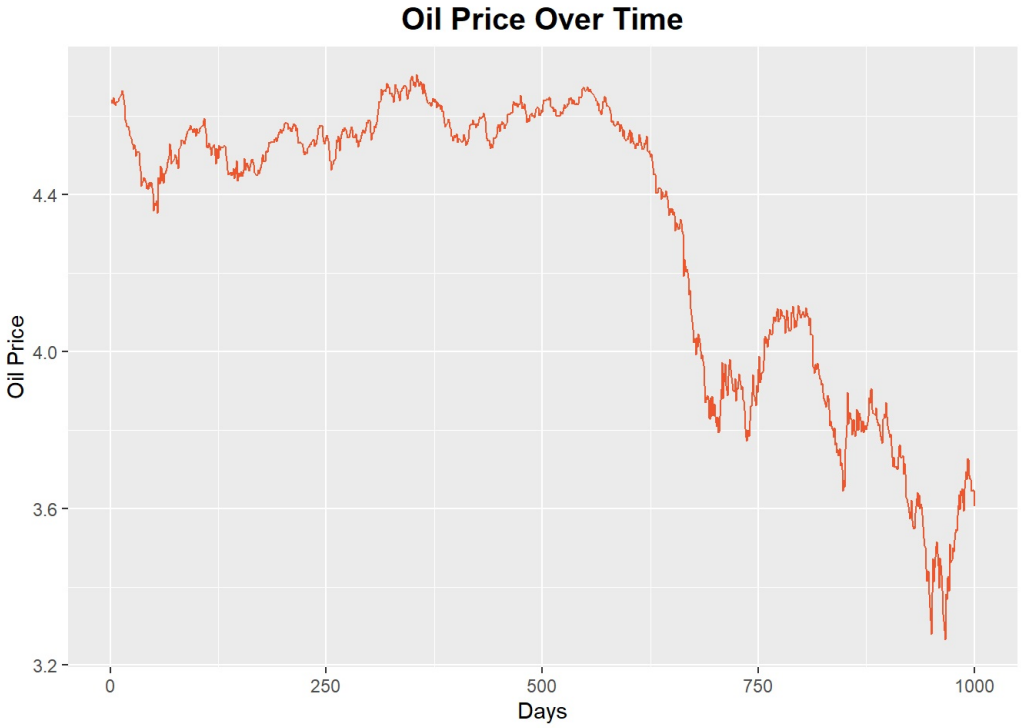
- Cooks Test for anomalies
- Test the response variable for stationarity
- Test the response variable for a normality
- Test the response variable for seasonality
- Adjust for seasonality, normality and stationarity

---

1. Explore the response variable

- Show the variation of response variable with time

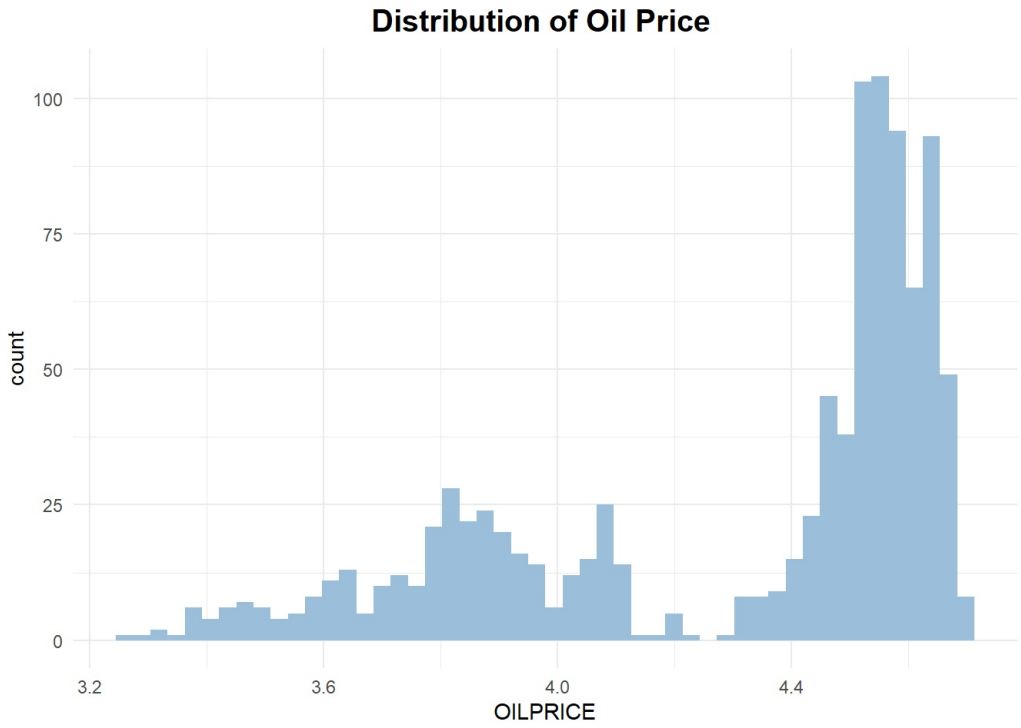
```
# Show trends of Oil Price with Time
ggplot(oil) +
 aes(x = index, y = OILPRICE) +
 geom_line(size = 0.5, colour = "#EF562D") +
 labs(title = 'Oil Price Over Time', x = 'Days', y = 'Oil Price') +
  theme(plot.title = element_text(size = 15L, face = "bold", hjust = 0.5))
```



Observation : We can roughly observe that the response variable trends negatively over the given time period

- Distribution of the response variable

```
ggplot(oil) +
 aes(x = OILPRICE) +
 geom_histogram(bins = 50L, fill = "#9BBEDB") +
 labs(title = "Distribution of Oil Price ") +
 theme_minimal() +
 theme(plot.title = element_text(size = 15L, face = "bold", hjust = 0.5))
```



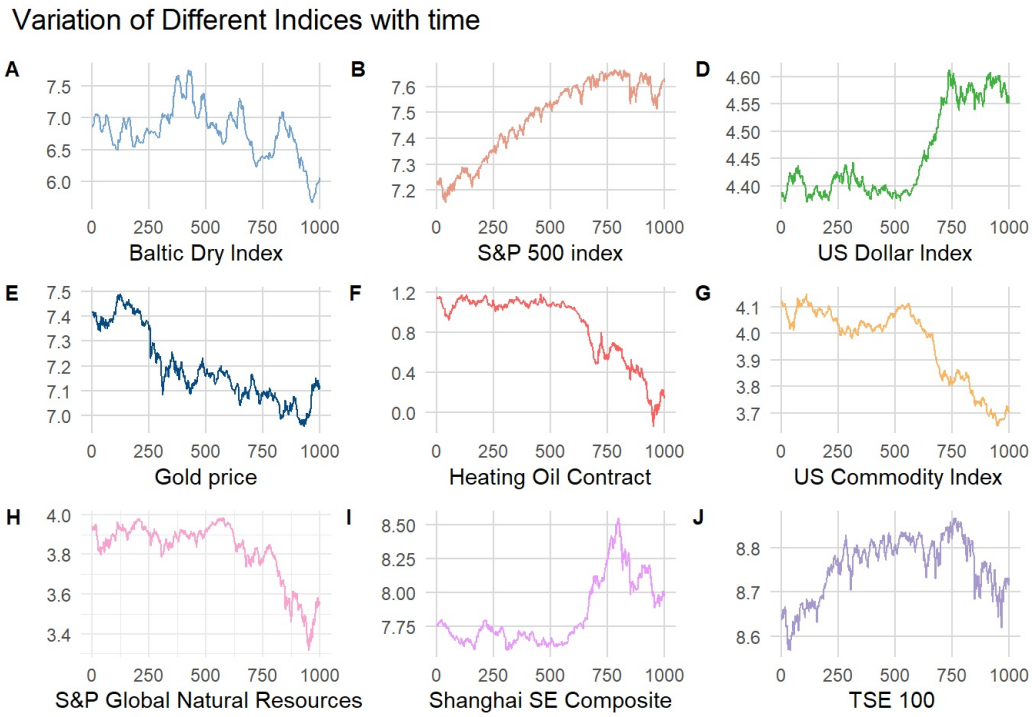
Observation : We can roughly observe a possible normal distribution skewed to the left

Explore the Covariates

- The trends of the covariates with time

```
p1 <- ggplot(oil) +
   aes(x = index, y = BDIY_log) +
 geom_line(size = 0.5, colour = "#71A1D0") +
  labs(x = "Baltic Dry Index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p2 <- ggplot(oil) +
   aes(x = index, y = SPX_log) +
 geom_line( size = 0.5, colour = "#E99984") +
  labs(x = "S&P 500 index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p3 <- ggplot(oil) +
  aes(x = index, y = DX1_log) +
 geom_line( size = 0.5, colour = "#3FB23F") +
  labs(x = "US Dollar Index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p4 <- ggplot(oil) +
   aes(x = index, y = GC1_log) +
 geom_line( size = 0.5, colour = "#064B84") +
  labs(x = "Gold price", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p5 <- ggplot(oil) +
   aes(x = index, y = HO1_log) +
 geom_line( size = 0.5, colour = "#FF5F5F") +
  labs(x = "Heating Oil Contract", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p6 <- ggplot(oil) +
   aes(x = index, y = USCI_log) +
 geom_line( size = 0.5, colour = "#FBB663") +
  labs(x = " US Commodity Index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p7 <- ggplot(oil) +
   aes(x = index, y = GNR_log) +
 geom_line( size = 0.5, colour = "#FBA3CF") +
  labs(x = " S&P Global Natural Resources", y ='') +
```

```
 theme_minimal()

p8 <- ggplot(oil) +
   aes(x = index, y = SHCOMP_log) +
 geom_line( size = 0.5, colour = "#EB99FF") +
  labs(x = " Shanghai SE Composite ", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p9 <- ggplot(oil) +
   aes(x = index, y = FTSE_log) +
 geom_line( size = 0.5, colour = "#A699CD") +
  labs(x = "TSE 100 ", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

# plot the grid
plot_row <-plot_grid(p1,
          p2,
          p3,
          p4,
          p5,
          p6,
          p7,
          p8,
          p9,
        labels = c('A', 'B', 'D', 'E', 'F', 'G', 'H', 'I', 'J'), label_size = 10, rel_heights = c(2, 2), align =
'hv' )

# now add the title
title <- ggdraw() +
  draw_label(
    "Variation of Different Indices with time",

    x = 0,
    hjust = 0
  ) +
  theme(
    # add margin on the left of the drawing canvas,
    # so title is aligned with left edge of first plot
    plot.margin = margin(0, 0, 0, 7)
  )
plot_grid(
  title, plot_row,
  ncol = 1,
  # rel_heights values control vertical title margins
  rel_heights = c(0.1, 1)
)
```

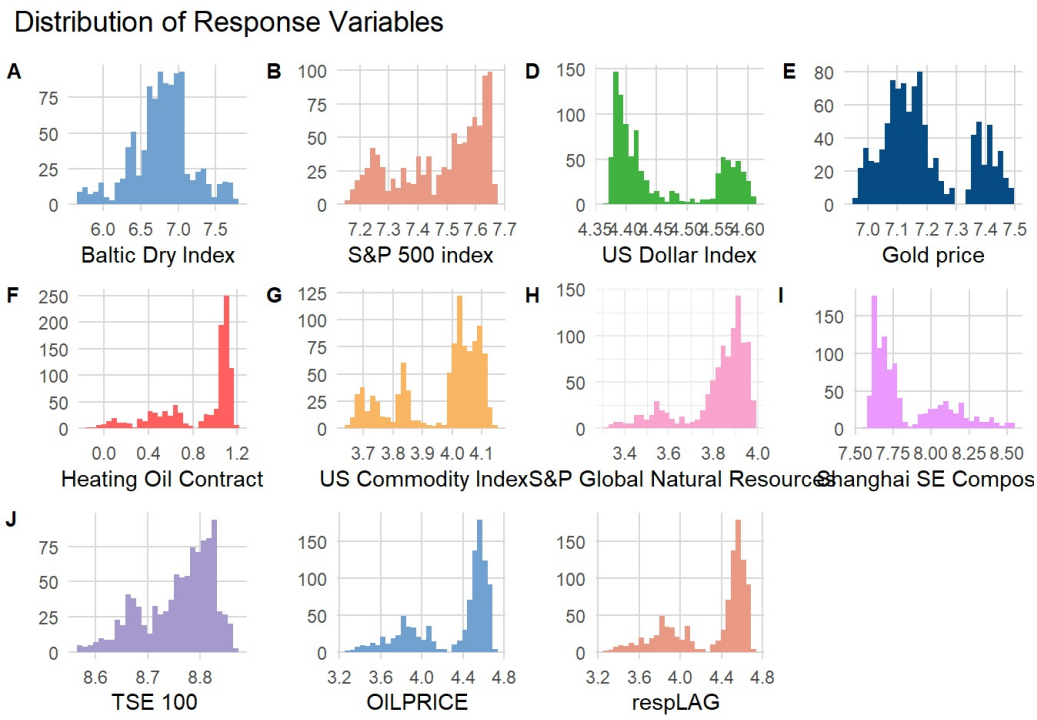


Observation : Visually we see A, E, H, F, G trend negatively with time and BDI trend positively with time. While J, visually seems to have no clear trend

- The distribution of the various response variables

```
# Check the distributions of all the variables

p1 <- ggplot(oil) +
  aes(x  = BDIY_log) +
 geom_histogram(bins = 30L, fill = "#71A1D0") +
  labs(x = "Baltic Dry Index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p2 <-ggplot(oil) +
 aes(x  = SPX_log) +
 geom_histogram(bins = 30L, fill = "#E99984") +
  labs(x = "S&P 500 index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

 p3 <-ggplot(oil) +
 aes(x  = DX1_log) +
 geom_histogram(bins = 30L, fill = "#3FB23F") +
  labs(x = "US Dollar Index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p4 <-ggplot(oil) +
 aes(x = GC1_log) +
 geom_histogram(bins = 30L, fill = "#064B84") +
  labs(x = "Gold price", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

 p5 <-ggplot(oil) +
 aes(x  = HO1_log) +
 geom_histogram(bins = 30L, fill = "#FF5F5F") +
  labs(x = "Heating Oil Contract", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p6 <-ggplot(oil) +
 aes(x  = USCI_log) +
 geom_histogram(bins = 30L, fill = "#FBB663") +
  labs(x = " US Commodity Index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p7 <-ggplot(oil) +
 aes(x  = GNR_log) +
 geom_histogram(bins = 30L, fill = "#FBA3CF") +
  labs(x = " S&P Global Natural Resources", y ='') +
 theme_minimal()

p8 <-ggplot(oil) +
 aes(x  = SHCOMP_log) +
 geom_histogram(bins = 30L, fill = "#EB99FF") +
  labs(x = " Shanghai SE Composite ", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p9 <-ggplot(oil) +
 aes(x = FTSE_log) +
 geom_histogram(bins = 30L, fill = "#A699CD") +
  labs(x = "TSE 100 ", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p10 <-ggplot(oil) +
 aes(x = OILPRICE) +
 geom_histogram(bins = 30L, fill = "#71A1D0") +
  labs(x = "OILPRICE ", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p11 <-ggplot(oil) +
 aes(x = respLAG) +
 geom_histogram(bins = 30L, fill = "#E99984") +
  labs(x = "respLAG ", y ='') +
 theme_minimal() +
```

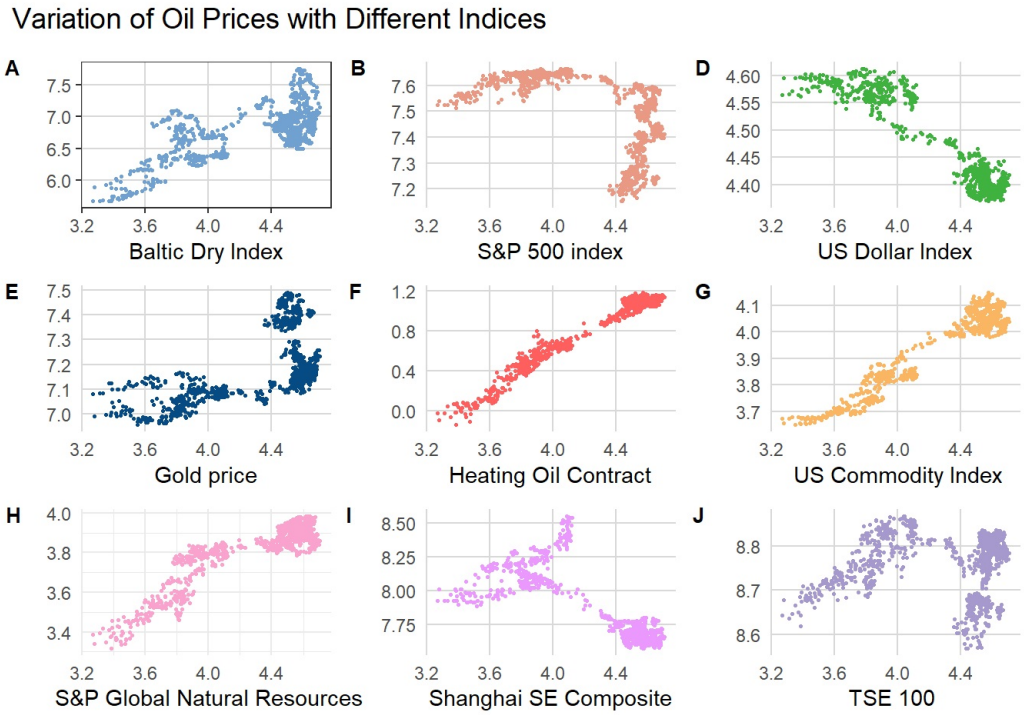
```
  background_grid(minor = 'none')

# plot the grid
plot_row <-plot_grid(p1,
          p2,
          p3,
          p4,
          p5,
          p6,
          p7,
          p8,
          p9, p10, p11,labels = c('A', 'B', 'D', 'E', 'F', 'G', 'H', 'I', 'J'), label_size = 10 )


title <- ggdraw() +
  draw_label(
    "Distribution of Response Variables",

    x = 0,
    hjust = 0
  ) +
  theme(
    # add margin on the left of the drawing canvas,
    # so title is aligned with left edge of first plot
    plot.margin = margin(0, 0, 0, 7)
  )
plot_grid(
  title, plot_row,
  ncol = 1,
  # rel_heights values control vertical title margins
  rel_heights = c(0.1, 1)
)
```



Observation : We see no clear indication of a normal distribution in these variables other than maybe the Baltic dry index

```
p1 <- ggplot(oil) +
  aes(x = OILPRICE, y = BDIY_log) +
 geom_point(shape = "circle", size = 0.5, colour = "#71A1D0") +
  labs(x = "Baltic Dry Index", y ='') +
 theme_bw() +
  background_grid(minor = 'none')

p2 <- ggplot(oil) +
 aes(x = OILPRICE, y = SPX_log) +
 geom_point(shape = "circle", size = 0.5, colour = "#E99984") +
  labs(x = "S&P 500 index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p3 <- ggplot(oil) +
 aes(x = OILPRICE, y = DX1_log) +
```

```
 geom_point(shape = "circle", size = 0.5, colour = "#3FB23F") +
  labs(x = "US Dollar Index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p4 <- ggplot(oil) +
 aes(x = OILPRICE, y = GC1_log) +
 geom_point(shape = "circle", size = 0.5, colour = "#064B84") +
  labs(x = "Gold price", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p5 <- ggplot(oil) +
 aes(x = OILPRICE, y = HO1_log) +
 geom_point(shape = "circle", size = 0.5, colour = "#FF5F5F") +
  labs(x = "Heating Oil Contract", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p6 <- ggplot(oil) +
 aes(x = OILPRICE, y = USCI_log) +
 geom_point(shape = "circle", size = 0.5, colour = "#FBB663") +
  labs(x = " US Commodity Index", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p7 <- ggplot(oil) +
 aes(x = OILPRICE, y = GNR_log) +
 geom_point(shape = "circle", size = 0.5, colour = "#FBA3CF") +
  labs(x = " S&P Global Natural Resources", y ='') +
 theme_minimal()

p8 <- ggplot(oil) +
 aes(x = OILPRICE, y = SHCOMP_log) +
 geom_point(shape = "circle", size = 0.5, colour = "#EB99FF") +
  labs(x = " Shanghai SE Composite ", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

p9 <- ggplot(oil) +
 aes(x = OILPRICE, y = FTSE_log) +
 geom_point(shape = "circle", size = 0.5, colour = "#A699CD") +
  labs(x = "TSE 100 ", y ='') +
 theme_minimal() +
  background_grid(minor = 'none')

# plot the grid
plot_row <-plot_grid(p1,
          p2,
          p3,
          p4,
          p5,
          p6,
          p7,
          p8,
          p9,
        labels = c('A', 'B', 'D', 'E', 'F', 'G', 'H', 'I', 'J'), label_size = 10, rel_heights = c(2, 2), align =
'hv' )

# now add the title
title <- ggdraw() +
  draw_label(
    "Variation of Oil Prices with Different Indices",

    x = 0,
    hjust = 0
  ) +
  theme(
    # add margin on the left of the drawing canvas,
    # so title is aligned with left edge of first plot
    plot.margin = margin(0, 0, 0, 7)
  )
plot_grid(
  title, plot_row,
  ncol = 1,
  # rel_heights values control vertical title margins
  rel_heights = c(0.1, 1)
)
```



Anomalies in the Variables

Prepare the data to test for anomalies

- Add a date variable to the data-set
- Group the data-set by time periods

- Add a date variable to the data-set

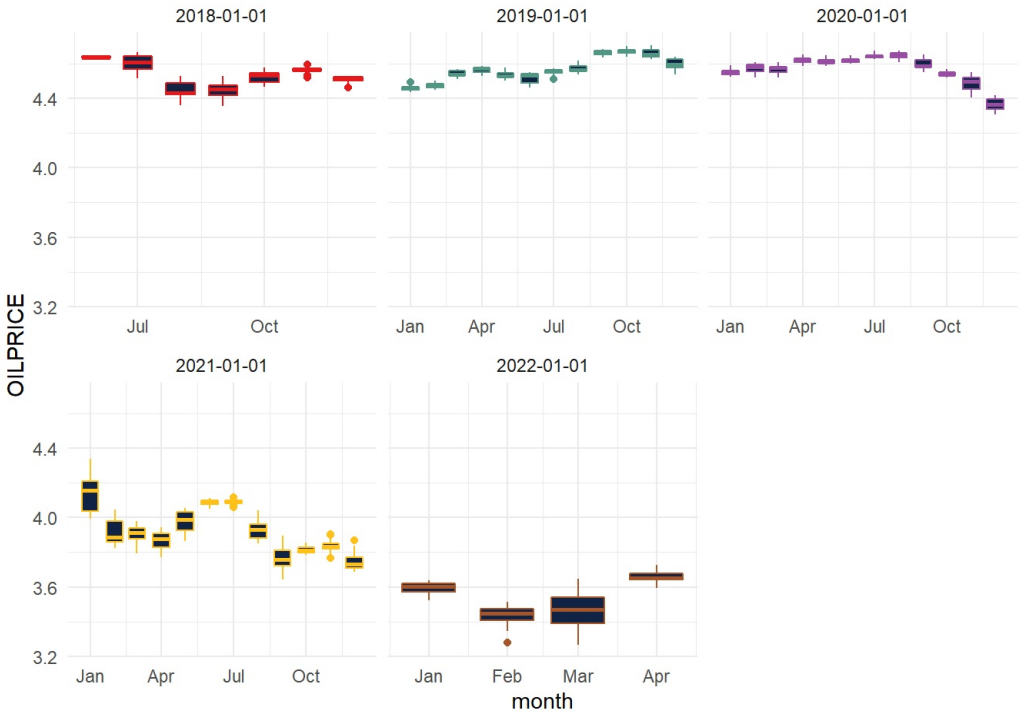
```
oil_transf <- oil[c("index","OILPRICE")]

# Both WTI and Brent Crude futures are traded from Sunday through Friday, 6:00 p.m. to 5:00 - So we add only Week days to the R table
x<-seq(as.Date("2018-01-01"),as.Date("2022-4-22"),by = 1)
x<-x[!weekdays(x) %in% c("Saturday","Sunday")]
df<-data.frame(x)

# Adding a Date column to the Table
oil_transf['Date'] <- df %>% slice(126:1125)
```

- Group the data-set by time periods

```
df <- data.frame(week = oil_transf['Date'],
                 OILPRICE = oil_transf['OILPRICE'])

# Add columns for Week, month and year
df['week'] <- floor_date(df$Date, "week")
df['month'] <- floor_date(df$Date, "month")
df['year'] <- floor_date(df$Date, "year")
```

- Plot anomalies in Monthly Data

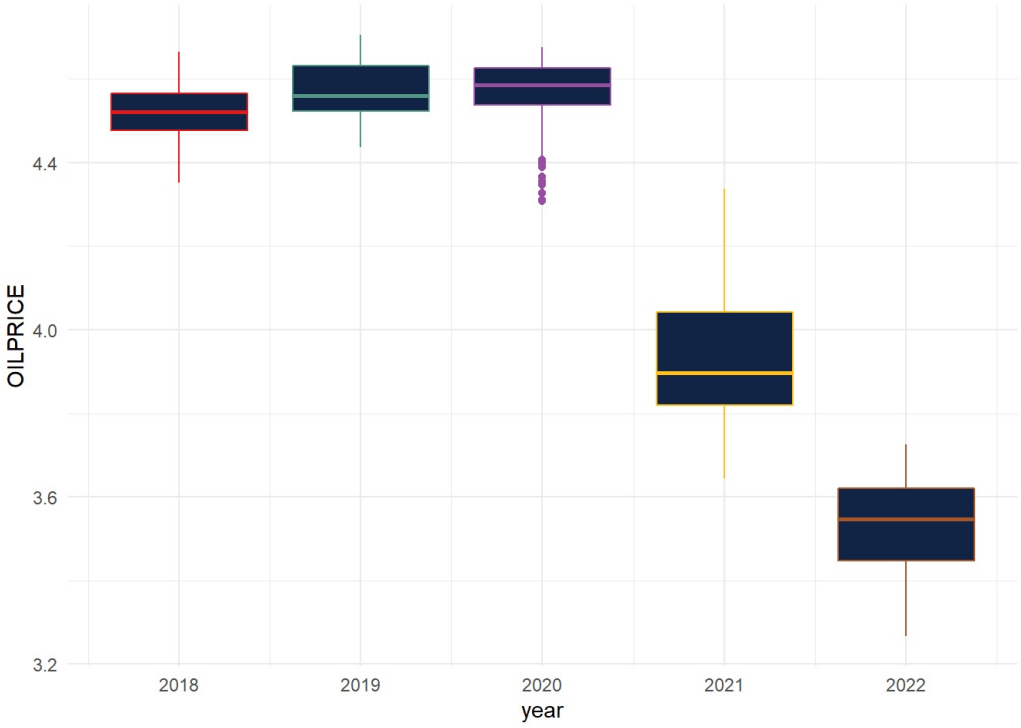
```
ggplot(df) +
 aes(x = month, y = OILPRICE, group = month, colour = year) +
 geom_boxplot(fill = "#112446") +
  scale_color_distiller(palette = "Set1", direction = 1) +
   scale_x_date(date_labels = "%b")+
 theme_minimal() +
 theme(legend.position = "none")+
  facet_wrap(vars(year), scales = "free_x")
```



Observations: The third quarter of the year shows more variations in oil prices.

- Anomalies in Yearly Data

```
ggplot(df) +
 aes(x = year, y = OILPRICE, group = year, colour = year) +
 geom_boxplot(fill = "#112446") +
  scale_color_distiller(palette = "Set1", direction = 1) +
 theme_minimal() +
 theme(legend.position = "none")
```



Observations: The year 2020 had most anomalies in the data and 2021 seems to have the greatest variations.

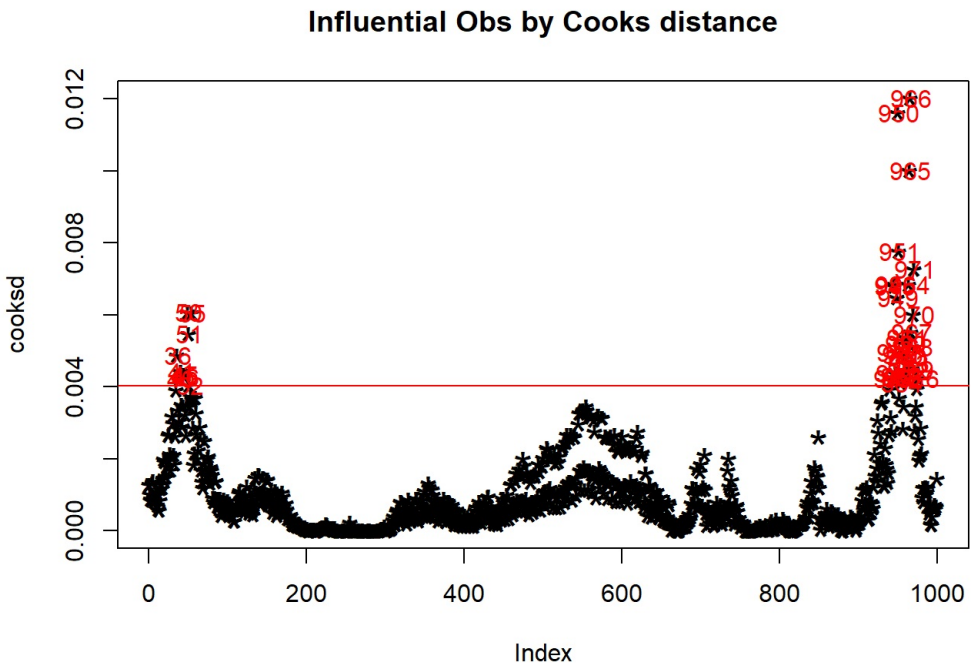
Part B : Statistical Analysis and Corrections

- Cooks test for anomalies
- Test the response variable for stationarity
- Test the response variable for a normality
- Test the response variable for seasonality
- Adjust for anomalies, seasonality, normality and stationarity

---

Cooks Test for Anomalies

```
mod <- lm(OILPRICE ~ ., data=oil_transf)
cooksd <- cooks.distance(mod)

plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")  # plot cook's distance
abline(h = 4*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4*mean(cooksd, na.rm=T),names(cooksd),""), col="red")
```



Observations : In general use, those observations that have a cook's distance greater than 4 times the mean may be classified as influential. This is not a hard boundary.

Show the influential rows :

```
influential <- as.numeric(names(cooksd)[(cooksd > 4*mean(cooksd, na.rm=T))])
head(oil_transf[influential, ])  # influential observations.
```

```
##    index OILPRICE       Date
## 36    36 4.421608 2018-08-13
## 40    40 4.440531 2018-08-17
## 41    41 4.432007 2018-08-20
## 45    45 4.429745 2018-08-24
## 46    46 4.431174 2018-08-27
## 50    50 4.359270 2018-08-31
```

These are the values that most affect the data set.

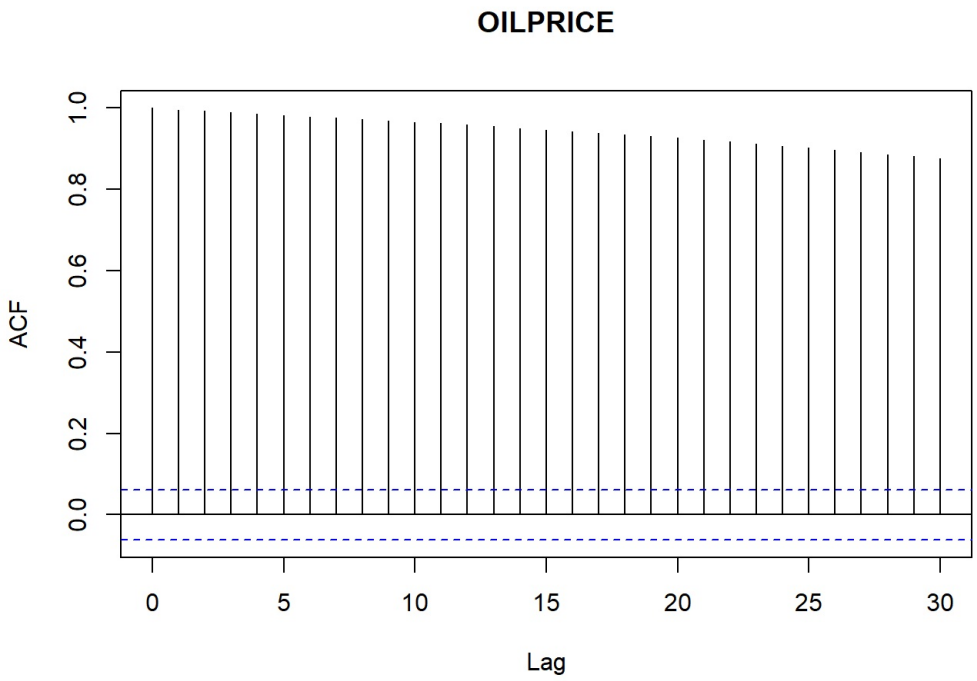
Test the response variable for stationarity

Reference: https://rpubs.com/richkt/269797 (https://rpubs.com/richkt/269797)

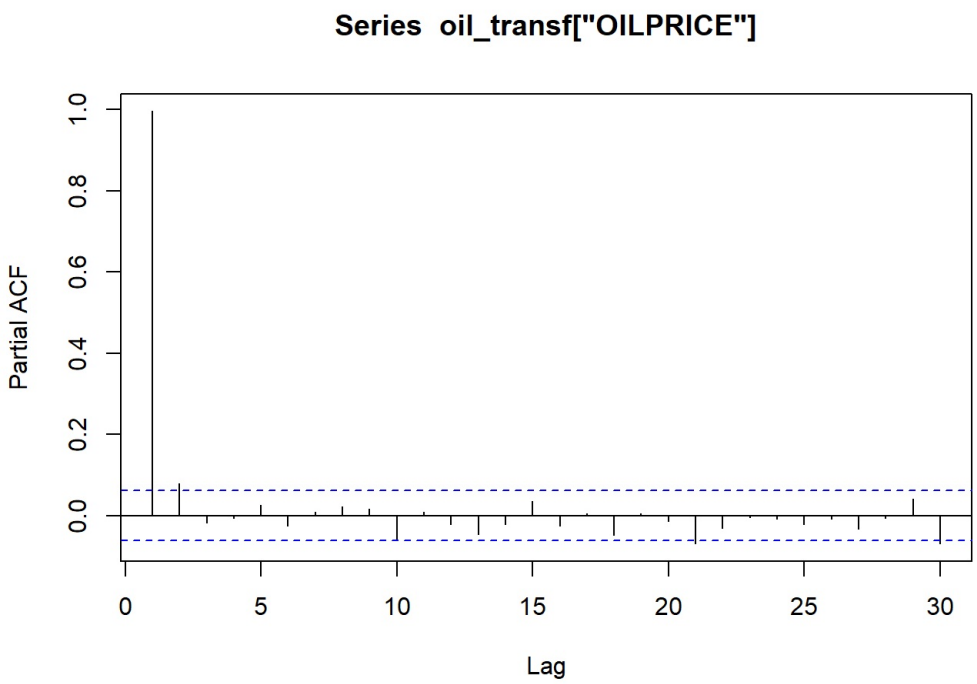
When investigating a time series, one of the first things to check before building an ARIMA model is to check that the series is stationary. That is, it needs to be determined that the time series is constant in mean and variance are constant and not dependent on time.

- All tests include :
    - Autocorrelation Function (ACF) (https://rstudio-pubs-static.s3.amazonaws.com/269797_8a8da7911da74ea38e2708a954aeaf29.html#autocorrelation-function-acf)
    - Ljung-Box test for independence (https://rstudio-pubs-static.s3.amazonaws.com/269797_8a8da7911da74ea38e2708a954aeaf29.html#ljung-box-test-for-independence)
    - Augmented Dickey–Fuller (ADF) t-statistic test for unit root (https://rstudio-pubs-static.s3.amazonaws.com/269797_8a8da7911da74ea38e2708a954aeaf29.html#augmented-dickeyfuller-adf-t-statistic-test-for-unit-root)
    - Kwiatkowski-Phillips-Schmidt-Shin (KPSS) for level or trend stationarity (https://rstudio-pubs-static.s3.amazonaws.com/269797_8a8da7911da74ea38e2708a954aeaf29.html#kwiatkowski-phillips-schmidt-shin-kpss-for-level-or-trend-stationarity)
- **ACF AND PACF**

```
# Plot the correaltion Function
autocor <- acf(oil_transf['OILPRICE'])
```




```
# Plot the autocorrealtion function
partialauto <-pacf(oil_transf['OILPRICE'])
```



Observation: From the graph we can see the the lag is significantly correlated with current series and thus the time series is not stationary

Augmented Dickey–Fuller (ADF)

```
tsData <- ts(oil_transf['OILPRICE'])
adf.test(tsData) # p-value < 0.05 indicates the TS is stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  tsData
## Dickey-Fuller = -1.4688, Lag order = 9, p-value = 0.8031
## alternative hypothesis: stationary
```

Observation : p-value < 0.05 indicates the TS is stationary. Since p is 0.8031, we can conclude that the data is not stationary

Kwiatkowski-Phillips-Schmidt-Shin (KPSS) for level or trend stationarity

```
kpss.test(tsData)
```

```
## Warning in kpss.test(tsData): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsData
## KPSS Level = 8.9101, Truncation lag parameter = 7, p-value = 0.01
```

Observation : Here we will test the null hypothesis of trend stationarity (a low *p*-value will indicate a signal that is not trend stationary, has a unit root)

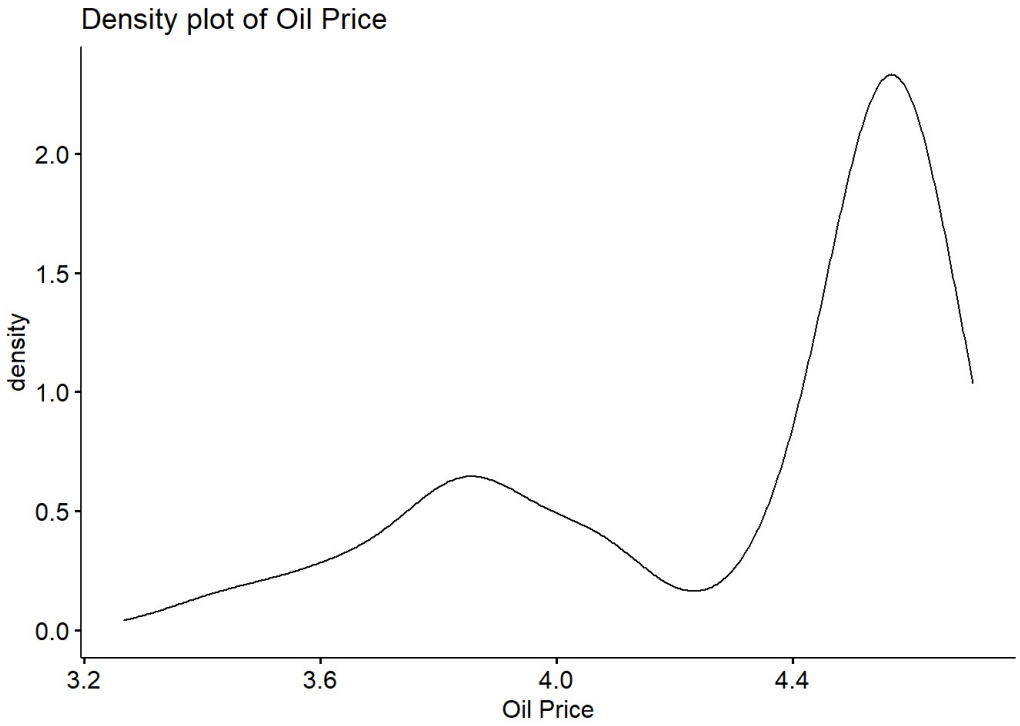
Test the response variable for a normality

Reference : http://www.sthda.com/english/wiki/normality-test-in-r (http://www.sthda.com/english/wiki/normality-test-in-r)

- Visual Method : Density Plot
- Q-Q Plot
- Statistical Tests : Normality Tests
    - Shapiro-Wilk's test
    - Kolmogorov-Smirnov (K-S) normality test (not v accurate)

Visual Method : Density Plot

```
ggdensity(oil_transf$OILPRICE,
          main = "Density plot of Oil Price",
          xlab = "Oil Price")
```

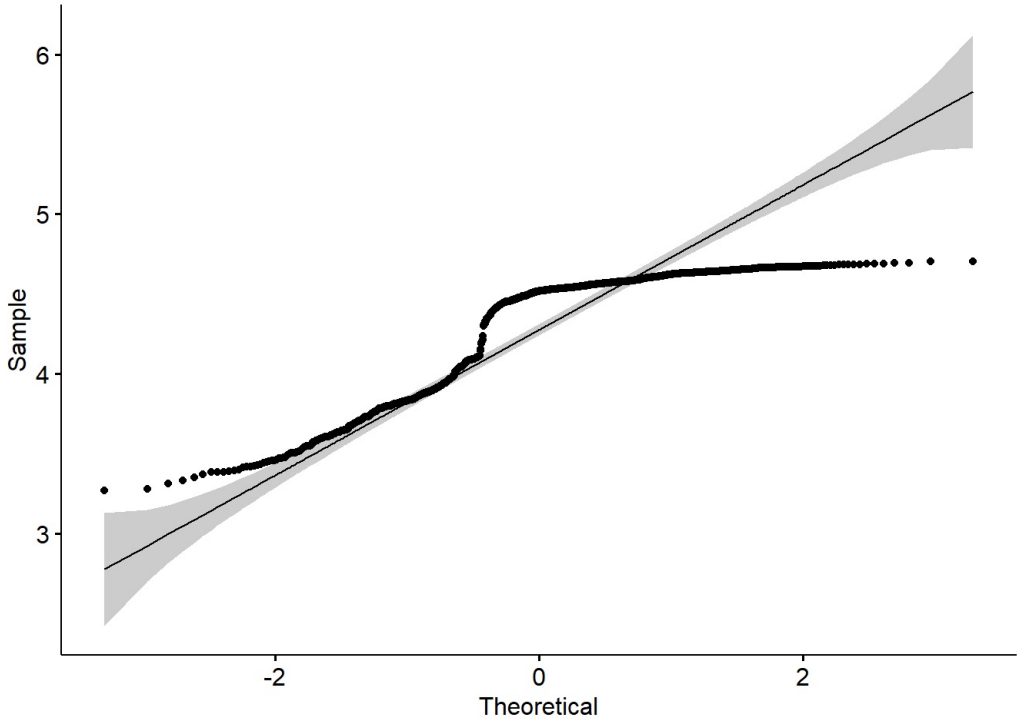


Observation : The distribution does not seem to be normal

Q-Q Plot

This plot draws the correlation between a given sample and the normal distribution. A 45-degree reference line is also plotted.

```
ggqqplot(oil_transf$OILPRICE)
```



Observation : The distribution does not seem to be normal

Normality Test Shapiro-Wilk's method is widely recommended for normality test and it provides better power than K-S. It is based on the correlation between the data and the corresponding normal scores.

```
shapiro.test(oil_transf$OILPRICE)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  oil_transf$OILPRICE
## W = 0.81804, p-value < 2.2e-16
```

Observation: From the output, the p-value < 0.05 implying that the distribution of the data are significantly different from normal distribution. In other words, we cannot assume normality

---

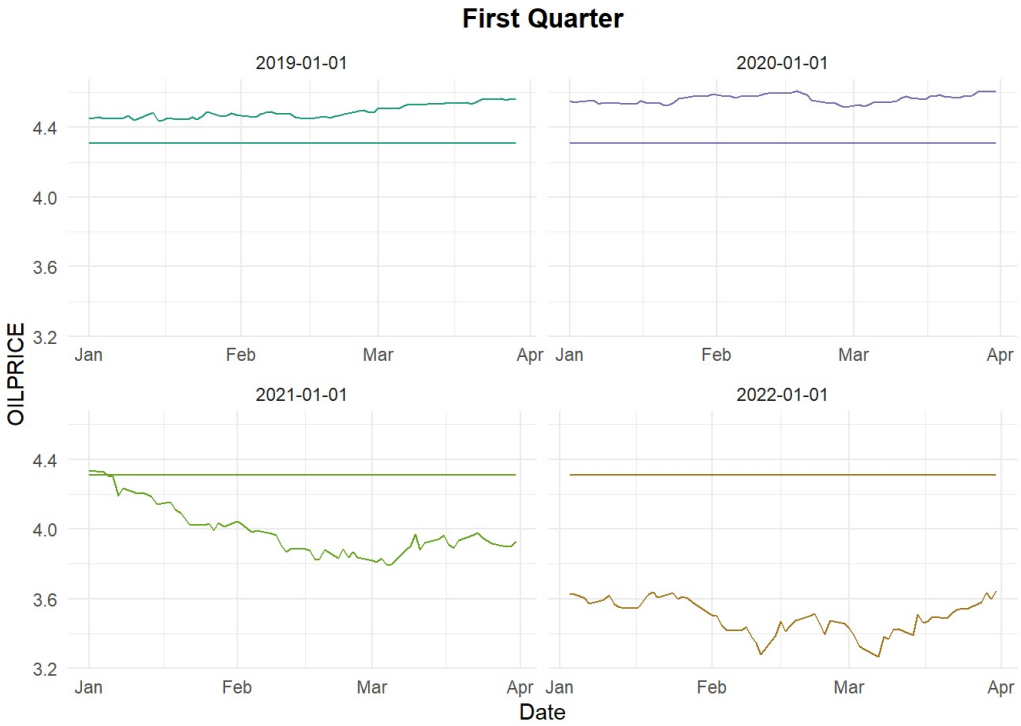
Test the response variable for seasonality

- Quarterly seasonality
- Decomposition of Seasonality

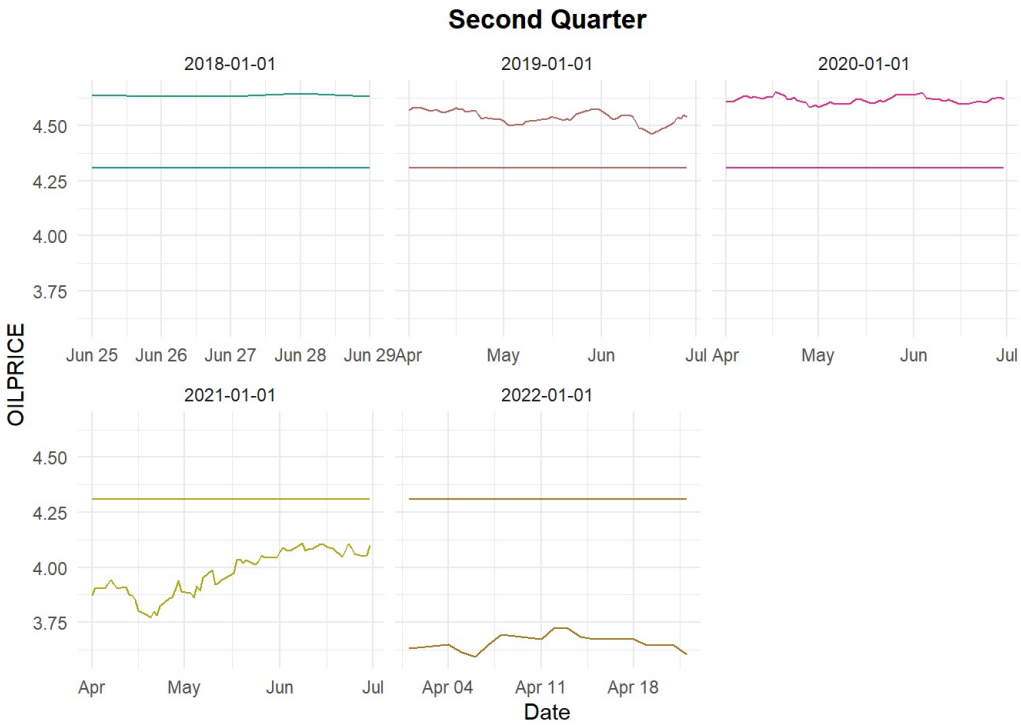
Quarterly Seasonality and Comparison with the Mean

```
# Prepare the seasonal data
first_quarter <- df %>%
    mutate(mean = mean(OILPRICE)) %>%
    filter(month(Date) <=3)

second_quarter <- df %>%
    mutate(mean = mean(OILPRICE)) %>%
    filter(month(Date) >3, month(Date) < 7)

third_quarter <- df %>%
    mutate(mean = mean(OILPRICE)) %>%
    filter(month(Date) >=7, month(Date) < 10)

fourth_quarter <- df %>%
    mutate(mean = mean(OILPRICE)) %>%
    filter(month(Date) >=10, month(Date) <= 12)
```

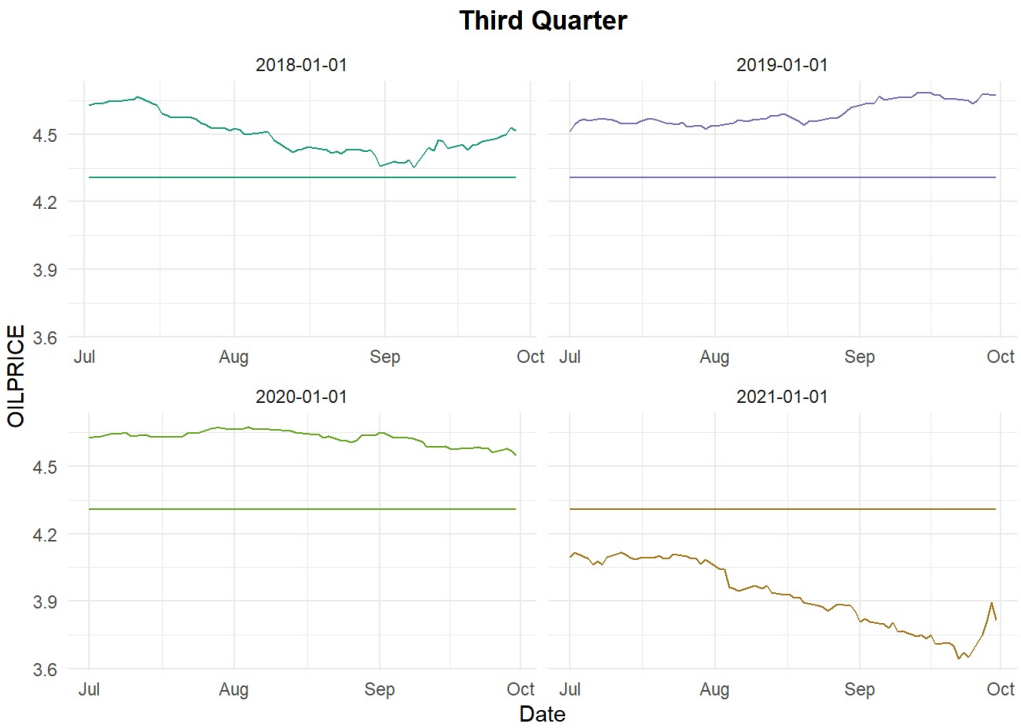
```
ggplot(first_quarter) +
  aes(x = Date, y = OILPRICE, colour = year) +
  geom_line(size = 0.5) +
  geom_line(aes(y=mean)) +
  scale_color_distiller(palette = "Dark2", direction = 1) +
  labs(title = "First Quarter") +
  theme_minimal() +
  theme(
    legend.position = "none",
    plot.title = element_text(face = "bold",
    hjust = 0.5)
  ) +
  facet_wrap(vars(year), scales = "free_x")
```



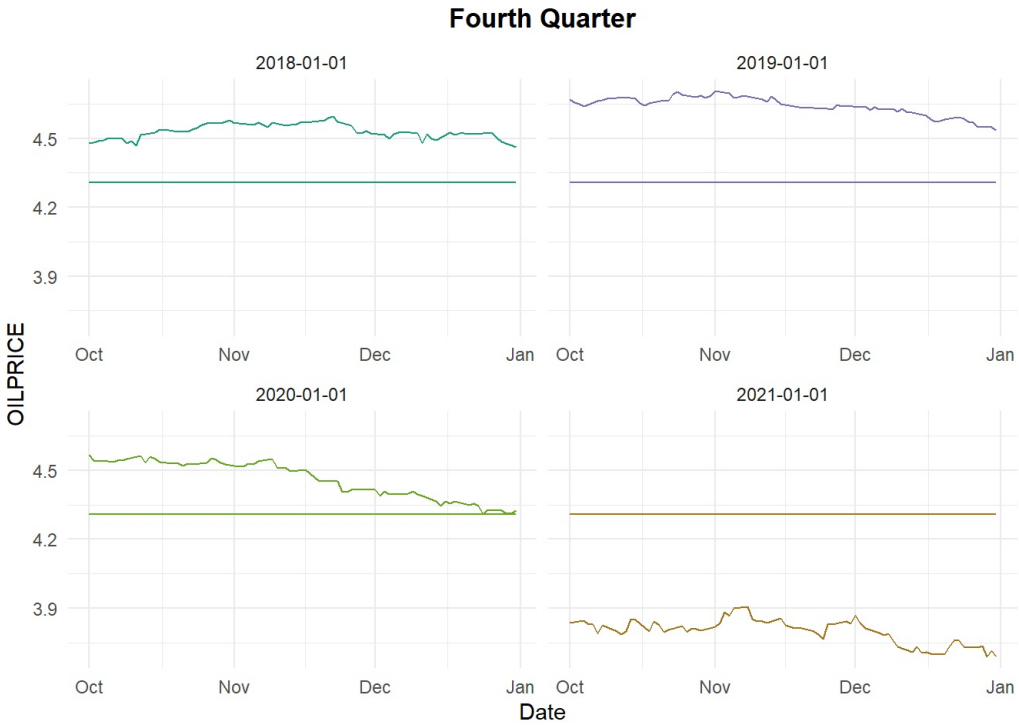

```
ggplot(second_quarter) +
  aes(x = Date, y = OILPRICE, colour = year) +
  geom_line(size = 0.5) +
  geom_line(aes(y=mean)) +
  scale_color_distiller(palette = "Dark2", direction = 1) +
  labs(title = "Second Quarter") +
  theme_minimal() +
  theme(
    legend.position = "none",
    plot.title = element_text(face = "bold",
    hjust = 0.5)
  ) +
  facet_wrap(vars(year), scales = "free_x")
```




```
ggplot(third_quarter) +
  aes(x = Date, y = OILPRICE, colour = year) +
  geom_line(size = 0.5) +
  geom_line(aes(y=mean)) +
  scale_color_distiller(palette = "Dark2", direction = 1) +
  labs(title = "Third Quarter") +
  theme_minimal() +
  theme(
    legend.position = "none",
    plot.title = element_text(face = "bold",
    hjust = 0.5)
  ) +
  facet_wrap(vars(year), scales = "free_x")
```



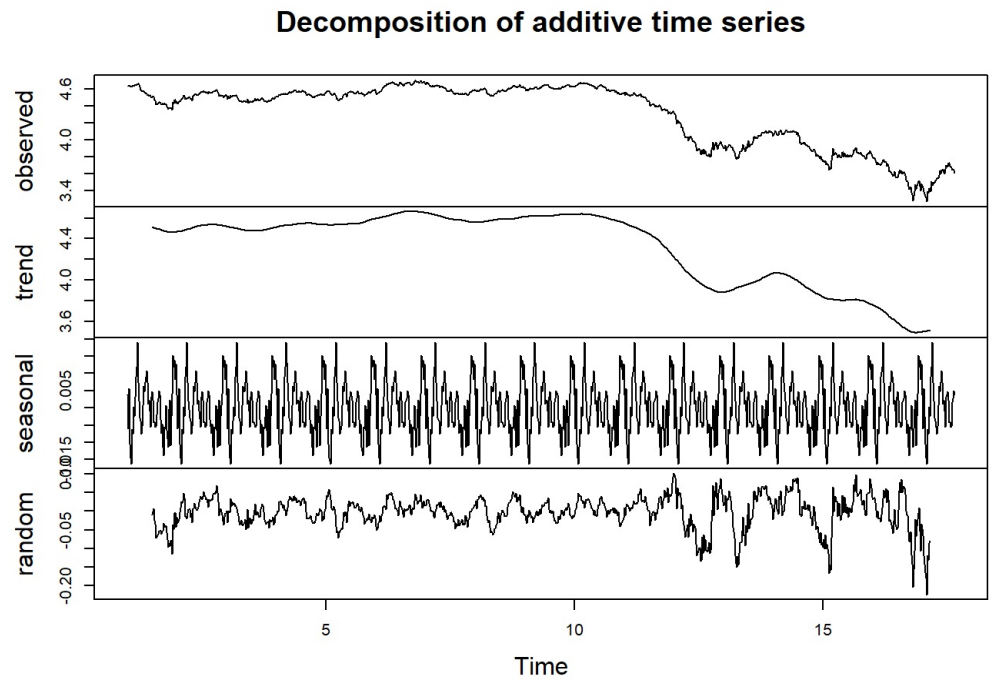
```
ggplot(fourth_quarter) +
  aes(x = Date, y = OILPRICE, colour = year) +
  geom_line(size = 0.5) +
  geom_line(aes(y=mean)) +
  scale_color_distiller(palette = "Dark2", direction = 1) +
  labs(title = "Fourth Quarter") +
  theme_minimal() +
  theme(
    legend.position = "none",
    plot.title = element_text(face = "bold",
    hjust = 0.5)
  ) +
  facet_wrap(vars(year), scales = "free_x")
```



Observations : The second quarter for all the years seem to have a lower than average oil price

Decomposition of Time Series

```
ts_data <- ts(as.vector(oil_transf['OILPRICE']), frequency = 60)

decomposedRes <- decompose(ts_data, type="additive")
plot (decomposedRes)
```



Adjust for anomalies, seasonality, normality and stationarity

Adjust for Anomalies - Not Needed

Reference : http://r-statistics.co/Outlier-Treatment-With-R.html (http://r-statistics.co/Outlier-Treatment-With-R.html)

Methods for treating outliers include :

1. Imputation : Imputation with mean / median / mode.
2. Capping : For missing values that lie outside the 1.5 * IQR limits, we could cap it by replacing those observations outside the lower limit with the value of 5th %ile and those that lie above the upper limit, with the value of 95th %ile.
3. Prediction : Outliers can be replaced with missing values (NA) and then can be predicted by considering them as a response variable

   Since oil prices tend to fluctuate to quite a massive extent and outliers may not be an anomaly, we let these outliers remain as part of the preditive model

Adjusting for Seasonality - Not Needed

Reference : http://r-statistics.co/Time-Series-Analysis-With-R.html (http://r-statistics.co/Time-Series-Analysis-With-R.html) De-seasonalizing throws insight about the seasonal pattern in the time series and helps to model the data without the seasonal effects.

Step 1: De-compose the Time series using `forecast::stl()`

Step 2: use `seasadj()` from 'forecast' package

Adjusting non-Stationarity

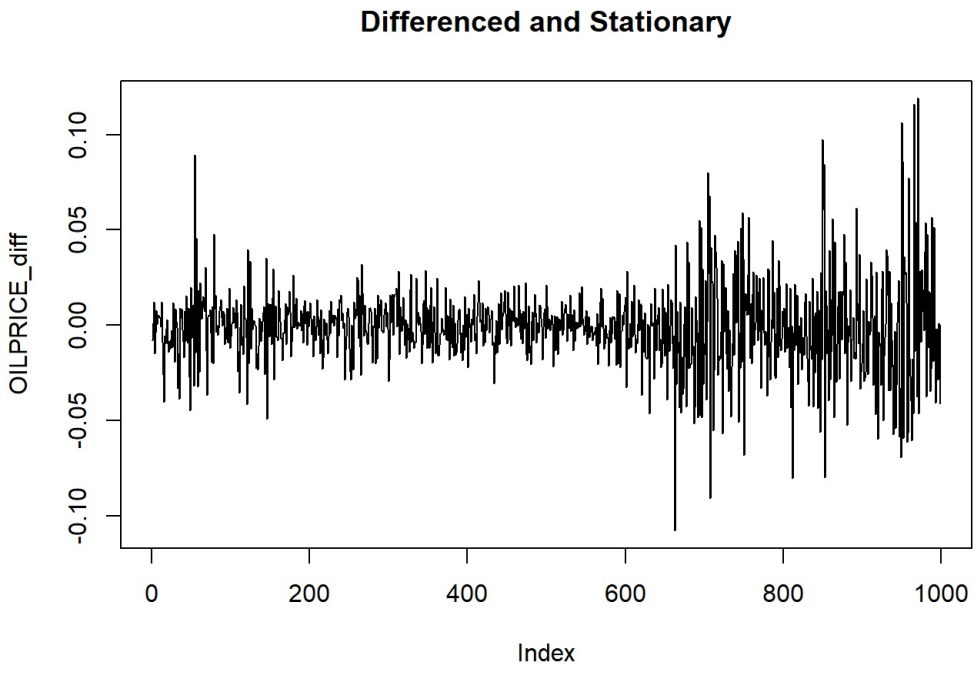
Reference : http://r-statistics.co/Time-Series-Analysis-With-R.html (http://r-statistics.co/Time-Series-Analysis-With-R.html)

Differencing a time series means, to subtract each data point in the series from its successor. It is commonly used to make a time series *stationary*. For most time series patterns, 1 or 2 differencing is necessary to make it a stationary series.

```
#  Differencing
ndiffs(oil_transf$OILPRICE)  # number for seasonal differencing needed
```

```
## [1] 1
```

```
#> 1
OILPRICE_diff <- diff(oil_transf$OILPRICE, lag=frequency(oil_transf$OILPRICE), differences=1)  # seasonal differe
ncing
plot(OILPRICE_diff, type="l", main="Differenced and Stationary")  # still not stationary!
```



```
# # Make it stationary
# ndiffs(OILPRICE_diff)  # number of differences need to make it stationary
# #> 1
# OILPRICE_diff <- diff(OILPRICE_diff, differences= 1)
# plot(stationaryTS, type="l", main="Differenced and Stationary")  # appears to be stationary
```

A regular differencing makes the time series stationary

Feature Selection and Feature Engineering

Reference : http://r-statistics.co/Variable-Selection-and-Importance-With-R.html (http://r-statistics.co/Variable-Selection-and-Importance-With-R.html) Finding the most important predictor variables (of features) that explains major part of variance of the response variable is key to identify and build high performing models.

Reference : https://www.datacamp.com/community/tutorials/feature-selection-R-boruta (https://www.datacamp.com/community/tutorials/feature-selection-R-boruta)

There are three types of feature selection methods in general:

- **Filter Methods** : filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithm. Instead the features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. Some common filter methods are Correlation metrics (Pearson, Spearman, Distance), Chi-Squared test, Anova, Fisher's Score etc.
- **Wrapper Methods** : in wrapper methods, you try to use a subset of features and train a model using them. Based on the inferences that you draw from the previous model, you decide to add or remove features from the subset. Forward Selection, Backward elimination are some of the examples for wrapper methods.
- **Embedded Methods** : these are the algorithms that have their own built-in feature selection methods. LASSO regression is one such example.

Methods :

- **Random Forest Method**
- **Relative Importance**
- **MARS**
- **Step-wise Regression**
- **Boruta**

- **Boruta**

```
# Boruta Search to find the best variables to use for prediction

# Decide if a variable is important or not using Boruta
boruta_output <- Boruta(OILPRICE~.-index, data = oil, doTrace=2)  # perform Boruta search
```

```
##  1. run of importance source...
```

```
##  2. run of importance source...
```

```
##  3. run of importance source...
```

```
##  4. run of importance source...
```

```
##  5. run of importance source...
```

```
##  6. run of importance source...
```

```
##  7. run of importance source...
```

```
##  8. run of importance source...
```

```
##  9. run of importance source...
```

```
##  10. run of importance source...
```

```
##  11. run of importance source...
```

```
##  12. run of importance source...
```

```
## After 12 iterations, +8.3 secs:
```

```
##  confirmed 20 attributes: CL10_log, CL11_log, CL12_log, CL13_log, CL14_log and 15 more;
```

```
##  still have 4 attributes left.
```

```
##  13. run of importance source...
```

```
##  14. run of importance source...
```

```
##  15. run of importance source...
```

```
##  16. run of importance source...
```

```
##  17. run of importance source...
```

```
##  18. run of importance source...
```

```
##  19. run of importance source...
```

```
##  20. run of importance source...
```

```
##  21. run of importance source...
```

```
##  22. run of importance source...
```

```
##  23. run of importance source...
```

```
##  24. run of importance source...
```

```
##  25. run of importance source...
```

```
##  26. run of importance source...
```

```
##  27. run of importance source...
```

```
##  28. run of importance source...
```

```
##  29. run of importance source...
```

```
##  30. run of importance source...
```

```
##  31. run of importance source...
```

```
##  32. run of importance source...
```

```
## After 32 iterations, +21 secs:
```

```
##  confirmed 1 attribute: SHCOMP_log;
```

```
##  still have 3 attributes left.
```

```
##  33. run of importance source...
```

```
##  34. run of importance source...
```

```
##  35. run of importance source...
```

```
##  36. run of importance source...
```

```
##  37. run of importance source...
```

```
##  38. run of importance source...
```

```
##  39. run of importance source...
```

```
##  40. run of importance source...
```

```
##  41. run of importance source...
```

```
##  42. run of importance source...
```

```
##  43. run of importance source...
```

```
##  44. run of importance source...
```

```
##  45. run of importance source...
```

```
##  46. run of importance source...
```

```
##  47. run of importance source...
```

```
##  48. run of importance source...
```

```
##  49. run of importance source...
```

```
##  50. run of importance source...
```

```
##  51. run of importance source...
```

```
##  52. run of importance source...
```

```
##  53. run of importance source...
```

```
##  54. run of importance source...
```

```
##  55. run of importance source...
```

```
##  56. run of importance source...
```

```
##  57. run of importance source...
```

```
##  58. run of importance source...
```

```
##  59. run of importance source...
```

```
##  60. run of importance source...
```

```
##  61. run of importance source...
```

```
##  62. run of importance source...
```

```
##  63. run of importance source...
```

```
##  64. run of importance source...
```

```
##  65. run of importance source...
```

```
##  66. run of importance source...
```

```
##  67. run of importance source...
```

```
##  68. run of importance source...
```

```
##  69. run of importance source...
```

```
##  70. run of importance source...
```

```
##  71. run of importance source...
```

```
## After 71 iterations, +47 secs:
```

```
##  confirmed 1 attribute: DX1_log;
```

```
##  still have 2 attributes left.
```

```
##  72. run of importance source...
```

```
##  73. run of importance source...
```

```
##  74. run of importance source...
```

```
##  75. run of importance source...
```

```
##  76. run of importance source...
```

```
##  77. run of importance source...
```

```
##  78. run of importance source...
```

```
##  79. run of importance source...
```

```
##  80. run of importance source...
```

```
##  81. run of importance source...
```

```
## After 81 iterations, +54 secs:
```

```
##  confirmed 1 attribute: GNR_log;
```

```
##  still have 1 attribute left.
```

```
##  82. run of importance source...
```

```
##  83. run of importance source...
```

```
##  84. run of importance source...
```

```
##  85. run of importance source...
```

```
##  86. run of importance source...
```

```
##  87. run of importance source...
```

```
##  88. run of importance source...
```

```
##  89. run of importance source...
```

```
##  90. run of importance source...
```

```
##  91. run of importance source...
```

```
##  92. run of importance source...
```

```
##  93. run of importance source...
```

```
##  94. run of importance source...
```

```
##  95. run of importance source...
```

```
##  96. run of importance source...
```

```
##  97. run of importance source...
```

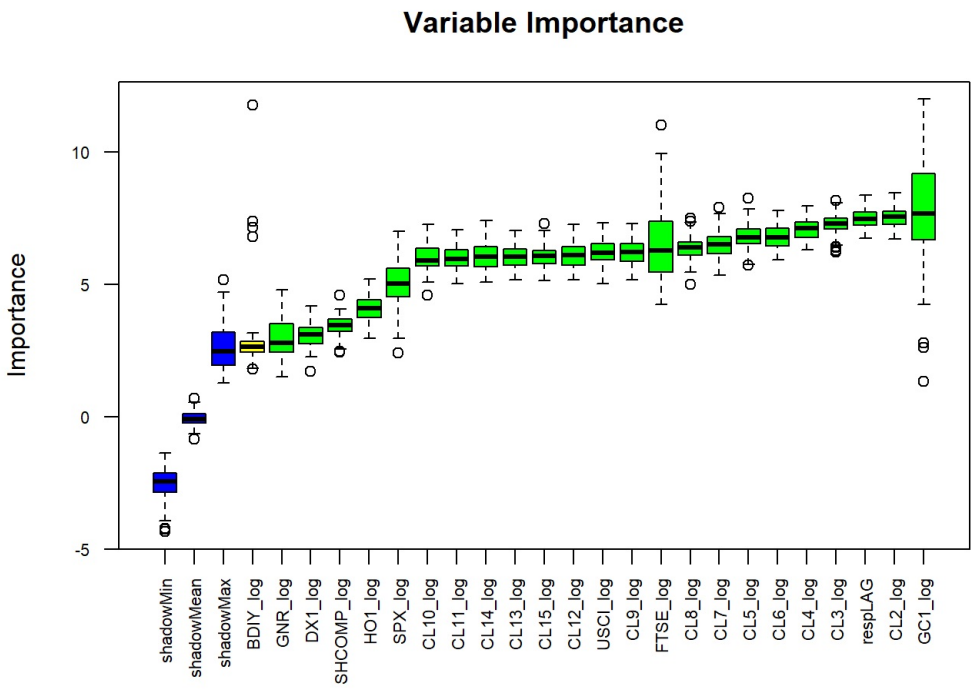
```
##  98. run of importance source...
```

```
##  99. run of importance source...
```

```
boruta_signif <- names(boruta_output$finalDecision[boruta_output$finalDecision %in% c("Confirmed", "Tentative")])
# collect Confirmed and Tentative variables
print(boruta_signif)  # significant variables
```

```
##  [1] "CL2_log"    "CL3_log"    "CL4_log"    "CL5_log"    "CL6_log"
##  [6] "CL7_log"    "CL8_log"    "CL9_log"    "CL10_log"   "CL11_log"
## [11] "CL12_log"   "CL13_log"   "CL14_log"   "CL15_log"   "BDIY_log"
## [16] "SPX_log"    "DX1_log"    "GC1_log"    "HO1_log"    "USCI_log"
## [21] "GNR_log"    "SHCOMP_log" "FTSE_log"   "respLAG"
```

```
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")  # plot variable importance
```



Thus the variables of importance are “CL2_log” “CL3_log” “CL4_log” “CL5_log” “CL6_log” “CL7_log” “CL8_log” “CL9_log” “CL10_log” “CL11_log” “CL12_log”

“CL13_log” “CL14_log” “CL15_log” “BDIY_log” “SPX_log” “DX1_log” “GC1_log” “HO1_log” “USCI_log” “GNR_log” “SHCOMP_log” “FTSE_log” and “respLAG”

We can group these into the following sets of models

1. Response variable as a function of all the future contract prices ie where the features are “CL2_log” “CL3_log” “CL4_log” “CL5_log” “CL6_log” “CL7_log” “CL8_log” “CL9_log” “CL10_log” “CL11_log” “CL12_log” “CL13_log” “CL14_log” “CL15_log”
2. Response variable as a function of the market indicies - “BDIY_log” “SPX_log” “DX1_log” “GC1_log” “HO1_log” “USCI_log” “GNR_log” “SHCOMP_log” “FTSE_log”
3. Response variable as a function of respLag
4. Response variable as a function of all of the above * This is the model implemented/tested below

Model Selection

The GAMLSS Model

GAMLSS : are distributional based semi-parametric regression type models.

- **regression type:** we have many explanatory variables X and one response variable y and we believe that X → y
- **distributional**: a full parametric distribution assumption is made for the response variable
- **semi-parametric**: All the parameters of the distribution can be modelled as function of the explanatory variables. Those functions can be parametric or non-parametric smoothing functions

GAMLSS philosophy: try different models

MODEL SELECTION STRATEGY

Let M = {D,L,T ,λ} represent a GAMLSS model .The components of M are defined as follows:

D specifies the distribution of the response variable,

L specifies the set of link functions for the distribution parameters μ,σ,ν and τ,

T specifies the terms appearing in the predictors for μ,σ,ν and τ,

λ specifies the smoothing hyper-parameters which determine the amount of smoothing of continuous explanatory variables and of the spatial effect.

In the search for an appropriate GAMLSS model for any new data set, all the above four components have to be specified as objectively as possible

Split the DataSet into Train and Test Samples

```
# Create a new dataframe based on selected parameters

oil_new <- oil %>%
  select("OILPRICE", "CL2_log"  ,  "CL3_log" ,   "CL4_log"  ,  "CL5_log"  ,  "CL6_log" ,   "CL7_log"  ,  "CL8_log
"  ,  "CL9_log"   , "CL10_log" ,  "CL11_log" ,  "CL12_log",

"CL13_log"  , "CL14_log" ,  "CL15_log"  , "BDIY_log",   "SPX_log"  ,  "DX1_log",    "GC1_log"   , "HO1_log" ,   "
USCI_log"  , "GNR_log"  ,  "SHCOMP_log", "FTSE_log" ,   "respLAG")
```

Split the data 80-20 to train and test

```
# Calculate the number of training rows
train_nrows <- 1000 * 0.80

# split training and test sets
train <- oil_new[1:train_nrows,]
test <- oil_new[(train_nrows + 1):1000,]
```

Step 1 : Selecting a Distribution

source : https://www.gamlss.com/short-courses-and-talks/ (https://www.gamlss.com/short-courses-and-talks/)

\reference (%5Bhttps://www.gamlss.com/short-courses-and-talks/reference)](https://www.gamlss.com/short-courses-and-talks/reference (https://www.gamlss.com/short-courses-and-talks/reference))) : chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.gamlss.com/wp-content/uploads/2013/01/gamlss-manual.pdf (extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.gamlss.com/wp-content/uploads/2013/01/gamlss-manual.pdf%5D(extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.gamlss.com/wp-content/uploads/2013/01/gamlss-manual.pdf))

The explicit gamlss.family distributions are subdivided in three distinct types according to the type of random variable modelled as a response.

Those three distinct types of GAMLSS distributions are:

1. **continuous** gamlss.family distributions,

2. **discrete** gamlss.family distributions and

3. **mixed** gamlss.family distributions.

We require the continuous distribtuions for our data.

The selection of the appropriate distribution D is done in two stages: the fitting stage and the diagnostic stage.

The **fitting stage** involves the comparison of different fitted models using a generalized Akaike information criterion (GAIC).

The **diagnostic stage** involves the normalized quantile residuals, or 'z-scores', which provide information about the adequacy of the model and can be used in connection with diagnostic plots like worm plots, or other test statistics, e.g. Z-statistics and Q-statistics.

The selection of the link function L is usually determined by the range of parameters. For a given distribution for the responsevariable,the selection of the terms T for the parameters of the distributions is done using a stepwise GAIC procedure.

```
mbest<-fitDist(train$OILPRICE,type="realline",k=2)
```

```
##
  |
  |                                                                      |   0%
  |
  |==                                                                    |   3%
  |
  |=====                                                                 |   7%
  |
  |=======                                                               |  10%
  |
  |==========                                                            |  14%
  |
  |============                                                          |  17%
  |
  |==============                                                        |  21%
  |
  |================                                                      |  24%
  |
  |====================                                                  |  28%
```

```
## Warning in MLE(ll3, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```

```
##
  |
  |======================                                                |  31%
```

```
## Warning in MLE(ll3, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```

```
##
  |
  |========================                                              |  34%
  |
  |==========================                                            |  38%
  |
  |============================                                          |  41%
```

```
## Warning in MLE(ll3, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```

```
##
  |
  |================================                                      |  45%
  |
  |=================================                                     |  48%
  |
  |==================================                                    |  52%
  |
  |====================================                                  |  55%
  |
  |=====================================                                 |  59%
  |
  |======================================                                |  62%
  |
  |=======================================                               |  66%
```

```
## Warning in MLE(ll4, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```

```
##
  |
  |========================================                              |  69%
  |
  |=========================================                             |  72%
```

```
## Warning in MLE(ll4, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```

```
##
  |
  |==========================================                            |  76%
```

```
## Warning in MLE(ll4, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```

```
##
  |
  |============================================                          |  79%
  |
  |=============================================                         |  83%
  |
  |==============================================                        |  86%
  |
  |===============================================                       |  90%
```

```
## Warning in MLE(ll4, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```

```
##
  |
  |================================================================      |  93%
  |
  |=================================================================     |  97%
  |
  |======================================================================| 100%
```

```
## Warning in MLE(ll4, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```

```
mbest
```

```
##
## Family:  c("SHASH", "Sinh-Arcsinh")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = y, family = DIST[i])
##
## Mu Coefficients:
## [1]  4.535
## Sigma Coefficients:
## [1]  9.152
## Nu Coefficients:
## [1]  9.804
## Tau Coefficients:
## [1]  11.59
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   796
## Global Deviance:     -901.978
##             AIC:     -893.978
##             SBC:     -875.24
```

```
print("The best fit distributions include : ")
```

```
## [1] "The best fit distributions include : "
```

```
mbest$fits
```

```
##      SHASH       SEP4    SHASHo2     SHASHo        JSU       JSUo        ST5
## -893.97819 -799.48012 -772.89096 -772.89096 -767.20577 -767.20577 -764.03679
##       EGB2        ST2        ST1       SEP3        ST3        SST        ST4
## -745.37774 -744.30300 -740.42889 -739.48047 -736.33393 -736.21380 -702.61872
##       SEP1       SEP2        SN2         GU         TF         GT        PE2
## -621.83860 -586.97534 -582.76304 -489.30799 -473.93193 -469.92502 -468.91850
##         PE        TF2        NET         LO         NO        SN1     exGAUS
## -468.37254 -407.83716 -325.47296 -154.03409  -61.81647  -59.81647  -59.71761
##         RG
##  334.57453
```

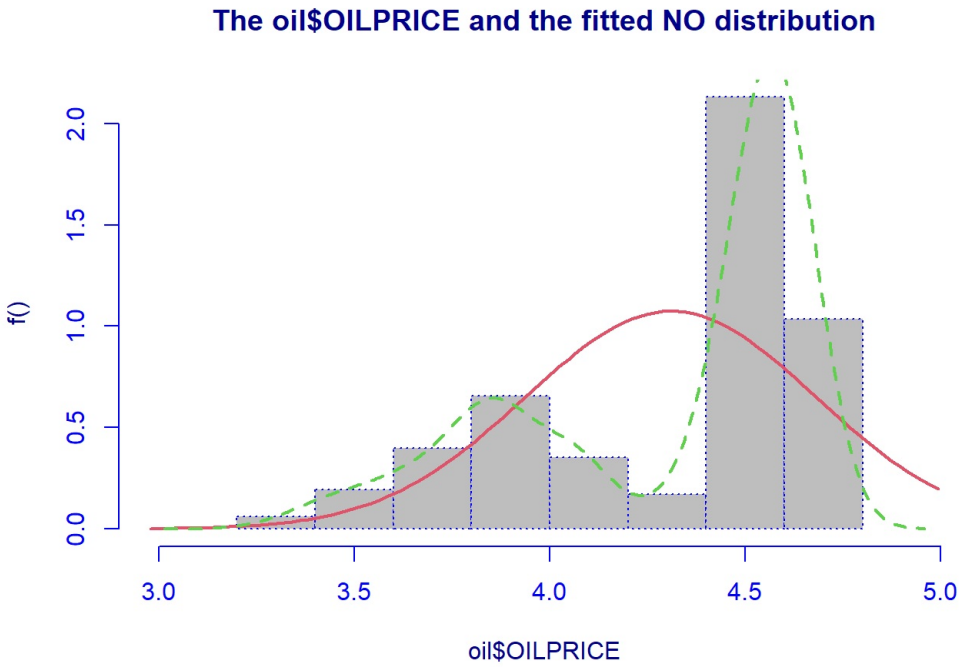
```
print("The Least suitable distributions include : ")
```

```
## [1] "The Least suitable distributions include : "
```

```
mbest$fails
```

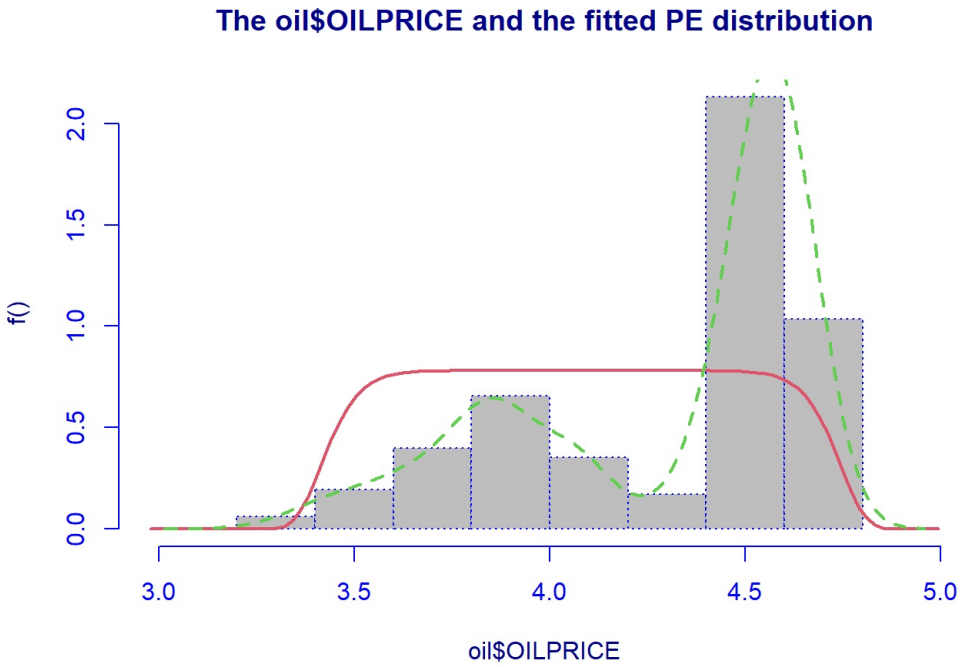
```
## NULL
```

```
# Find the appropriate distribution - Increasing Complexity in each distribution}
#------------------------------------------------------------------------------
# Normal Method
histDist(oil$OILPRICE,
         family = NO ,
         density = TRUE)
```




```
##
## Family:  c("NO", "Normal")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = oil$OILPRICE, family = "NO")
##
## Mu Coefficients:
## [1]  4.309
## Sigma Coefficients:
## [1]  -0.9908
##
##  Degrees of Freedom for the fit: 2 Residual Deg. of Freedom   998
## Global Deviance:     856.24
##             AIC:     860.24
##             SBC:     870.055
```

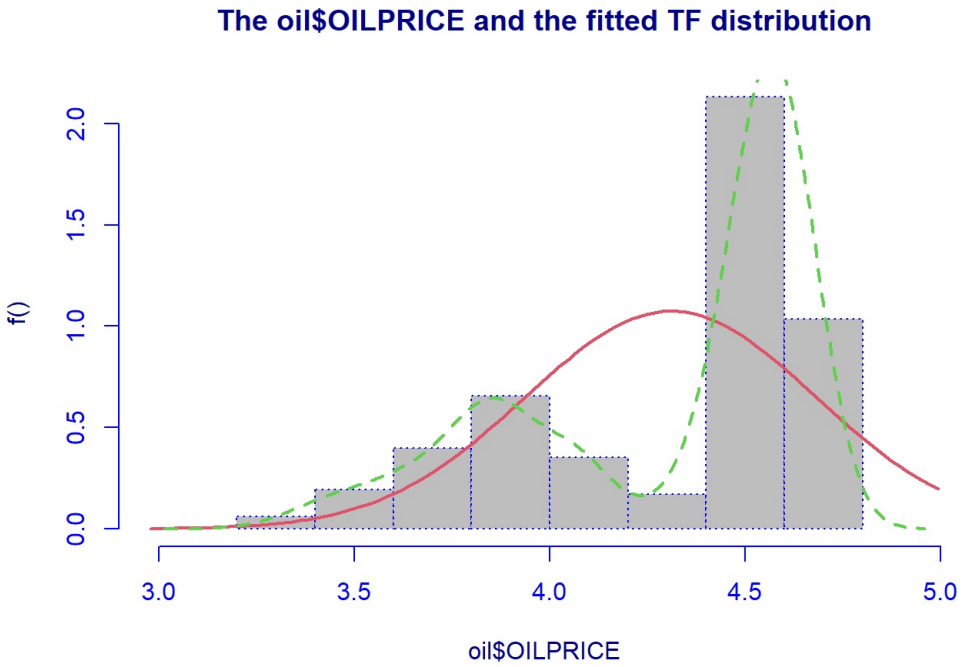
```
# Global Deviance:     856.24
#             AIC:     860.24
#             SBC:     870.055

#------------------------------------------------------------------------------
# Power Exponential Distribution
histDist(oil$OILPRICE,
         family = PE ,
         density = TRUE)
```




```
##
## Family:  c("PE", "Power Exponential")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = oil$OILPRICE, family = "PE")
##
## Mu Coefficients:
## [1]  4.079
## Sigma Coefficients:
## [1]  -0.9781
## Nu Coefficients:
## [1]  2.433
##
##  Degrees of Freedom for the fit: 3 Residual Deg. of Freedom   997
## Global Deviance:     671.657
##             AIC:     677.657
##             SBC:     692.38
```

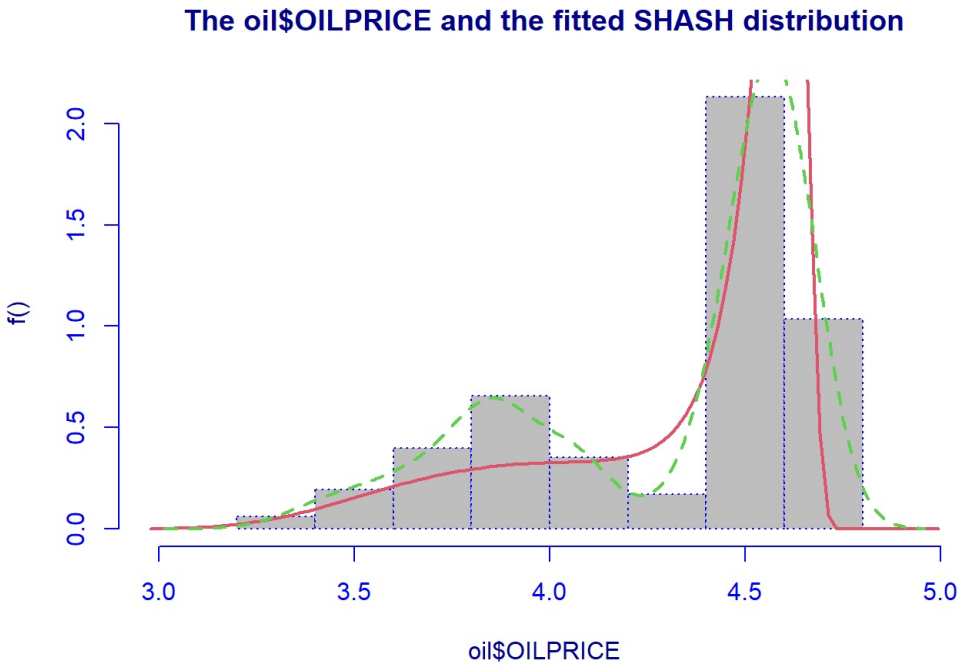
```
# Global Deviance:     671.657
#             AIC:     677.657
#             SBC:     692.38

#------------------------------------------------------------------------------
# t - distribution
histDist(oil$OILPRICE,
         family = TF ,
         density = TRUE)
```

```
## Error in solve.default(oout$hessian) :
##   Lapack routine dgesv: system is exactly singular: U[3,3] = 0
```




```
##
## Family:  c("TF", "t Family")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = oil$OILPRICE, family = "TF")
##
## Mu Coefficients:
## [1]  4.309
## Sigma Coefficients:
## [1]  -0.9908
## Nu Coefficients:
## [1]  13.87
##
##  Degrees of Freedom for the fit: 3 Residual Deg. of Freedom   997
## Global Deviance:     856.24
##             AIC:     862.24
##             SBC:     876.963
```

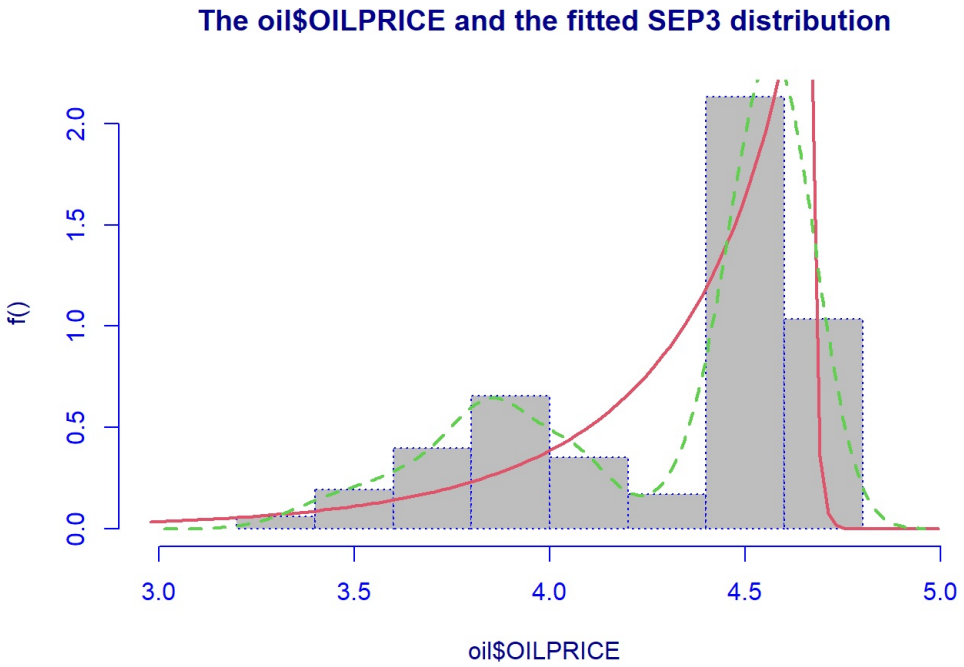
```
#  Degrees of Freedom for the fit: 3 Residual Deg. of Freedom   997
# Global Deviance:     856.24
#             AIC:     862.24
#             SBC:     876.963

#------------------------------------------------------------------------------
# Sinh-Arcsinh
histDist(oil$OILPRICE,
         family = SHASH ,
         density = TRUE)
```




```
##
## Family:  c("SHASH", "Sinh-Arcsinh")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = oil$OILPRICE, family = "SHASH")
##
## Mu Coefficients:
## [1]  4.524
## Sigma Coefficients:
## [1]  10.87
## Nu Coefficients:
## [1]  11.15
## Tau Coefficients:
## [1]  13.24
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   996
## Global Deviance:     -232.604
##             AIC:     -224.604
##             SBC:     -204.973
```

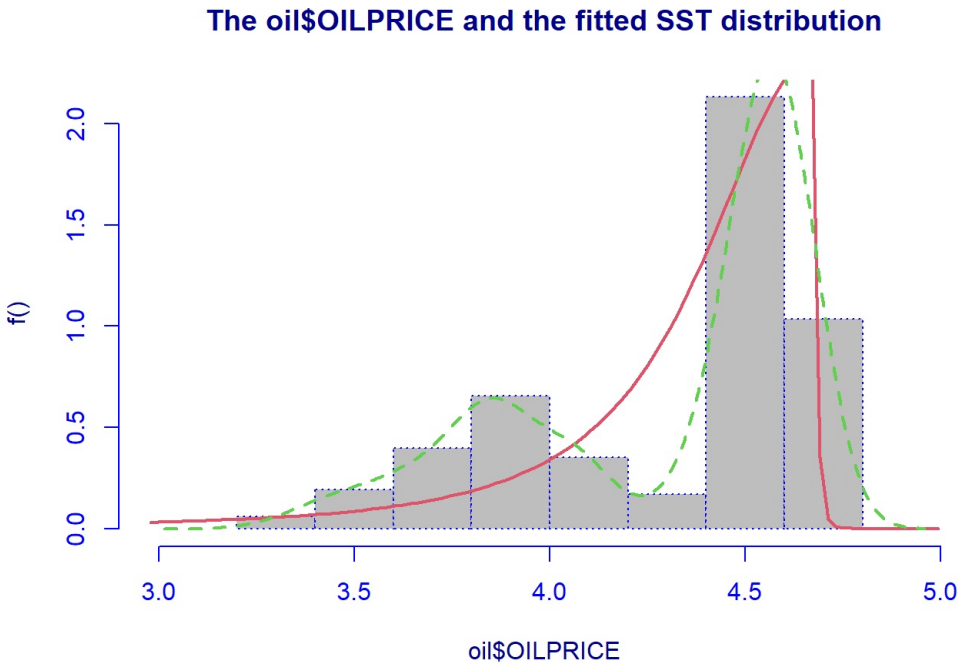
```
# Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   996
# Global Deviance:     -232.604
#             AIC:     -224.604
#             SBC:     -204.973

#------------------------------------------------------------------------------
#!!! skew exponential power type 3 !!!!!
histDist(oil$OILPRICE,
         family = SEP3 ,
         density = TRUE)
```

```
## Warning in MLE(ll4, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```




```
##
## Family:  c("SEP3", "skew exponential power type 3")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = oil$OILPRICE, family = "SEP3")
##
## Mu Coefficients:
## [1]  4.668
## Sigma Coefficients:
## [1]  -3.911
## Nu Coefficients:
## [1]  -1.718
## Tau Coefficients:
## [1]  -0.2067
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   996
## Global Deviance:     69.4867
##             AIC:     77.4867
##             SBC:     97.1177
```

```
#  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   996
# Global Deviance:     69.4867
#             AIC:     77.4867
#             SBC:     97.1177

#------------------------------------------------------------------------------
# SST
histDist(oil$OILPRICE,
         family = SST ,
         density = TRUE)
```

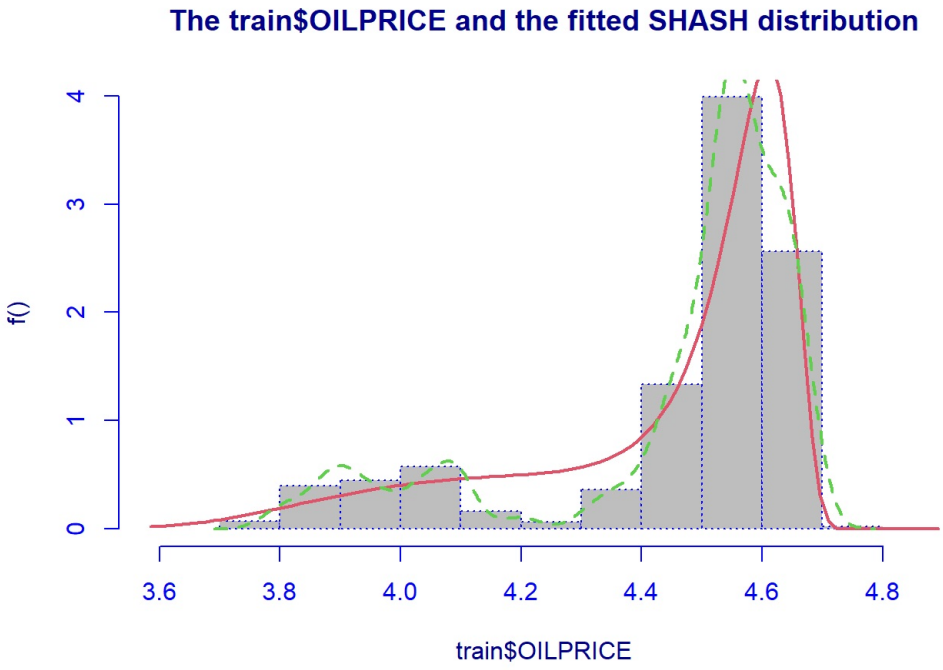



```
##
## Family:  c("SST", "SST")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = oil$OILPRICE, family = "SST")
##
## Mu Coefficients:
## [1]  4.265
## Sigma Coefficients:
## [1]  8.084
## Nu Coefficients:
## [1]  -1.847
## Tau Coefficients:
## [1]  -17.93
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   996
## Global Deviance:     179.463
##             AIC:     187.463
##             SBC:     207.094
```

```
#  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   996
# Global Deviance:     179.463
#             AIC:     187.463
#             SBC:     207.094
```

We see that of the distributions tested the SHASH and the skew exponential type distribution is the most apprpriate since it has the lowest GD,AIC,SBC. Three-parameter distributions are able to model either skewness or kurtosis in addition to location and scale We can test another distribution of the same family but a higher complexity(higher degree of freedom)

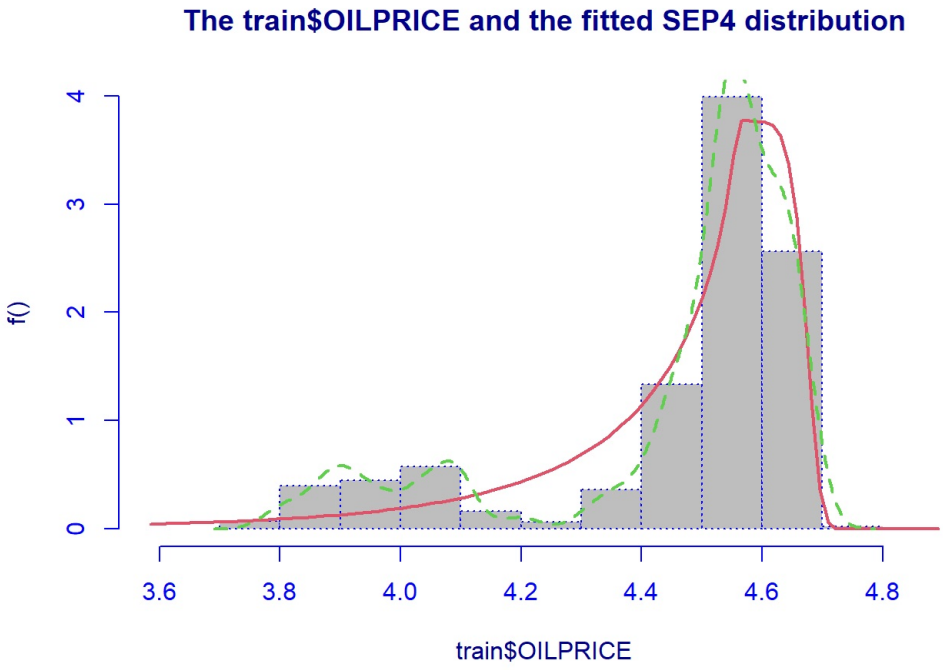
```
# Sinh-Arcsinh
histDist(train$OILPRICE,

         family = SHASH,

         density = T)
```




```
##
## Family:  c("SHASH", "Sinh-Arcsinh")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = train$OILPRICE, family = "SHASH")
##
## Mu Coefficients:
## [1]  4.535
## Sigma Coefficients:
## [1]  9.152
## Nu Coefficients:
## [1]  9.804
## Tau Coefficients:
## [1]  11.59
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   796
## Global Deviance:     -901.978
##             AIC:     -893.978
##             SBC:     -875.24
```

```
# Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   796
# Global Deviance:     -901.978
#             AIC:     -893.978
#             SBC:     -875.24
# skew exponential power type 4
histDist(train$OILPRICE,

         family = SEP4,

         density = T)
```

```
## Warning in MLE(ll4, start = list(eta.mu = eta.mu, eta.sigma = eta.sigma, :
## possible convergence problem: optim gave code=1 false convergence (8)
```




```
##
## Family:  c("SEP4", "skew exponential power type 4")
## Fitting method: "nlminb"
##
## Call:  gamlssML(formula = train$OILPRICE, family = "SEP4")
##
## Mu Coefficients:
## [1]  4.556
## Sigma Coefficients:
## [1]  -2.098
## Nu Coefficients:
## [1]  -0.3275
## Tau Coefficients:
## [1]  1.878
##
##  Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   796
## Global Deviance:     -807.48
##             AIC:     -799.48
##             SBC:     -780.742
```

```
# Degrees of Freedom for the fit: 4 Residual Deg. of Freedom   796
# Global Deviance:     -807.48
#             AIC:     -799.48
#             SBC:     -780.742
```

Select the Link Functions

Reference: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.econ-jobs.com/research/10106-Modelling-skewness-and-kurtosis-with-the-bcpe-density-in-gamlss.pdf (extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.econ-jobs.com/research/10106-Modelling-skewness-and-kurtosis-with-the-bcpe-density-in-gamlss.pdf)

The selection of the link function is usually determined by the range of parameters in hand. For example, in the Pareto II distribution, both μ and σ take values in the positive line, so a log link function seems a natural way of ensuring that they remain positive.

Select the terms in the Model

Reference : chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.econ-jobs.com/research/10106-Modelling-skewness-and-kurtosis-with-the-bcpe-density-in-gamlss.pdf (extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.econ-jobs.com/research/10106-Modelling-skewness-and-kurtosis-with-the-bcpe-density-in-gamlss.pdf)

Let X be the selection of all terms available for consideration. X could contain both linear and smoothing terms. For example, let f1 and f2 represent factors and x1, x2, x3 and x4 represent continuous explanatory variables, respectively. Then, X ={f1 ∗ f2 + s(x1) + s(x2) + s(x3) + x4} allows second-order interactions for the factors and smooth functions for x1, x2 and x3 and a linear term for x4.

For a given distribution for the response variable,the selection of the term sfor all the parameters of the distributions is done using a stepwise GAIC procedure. There are many different strategies that could be applied for the selection of the terms used to model the four parameters μ, σ, ν and τ.

Selection of Smoothing Parameters

Reference : chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.econ-jobs.com/research/10106-Modelling-skewness-and-kurtosis-with-the-bcpe-density-in-gamlss.pdf (extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.econ-jobs.com/research/10106-Modelling-skewness-and-kurtosis-with-the-bcpe-density-in-gamlss.pdf)

The following are three of the different methods of estimating the smoothing parameters:

1. using generalised cross validation

2. using GAIC

3. using local maximum-likelihood method or a PQL method.

Selection of Algorithm

The available algorithms include :

- CG
- RS
- Mixed

Without Additive Terms

```
m1.cg =gamlss(OILPRICE ~., family = SEP4,
      data = train, method=CG(50))
```

```
## GAMLSS-CG iteration 1: Global Deviance = -2683.764
## GAMLSS-CG iteration 2: Global Deviance = 10519.8
## GAMLSS-CG iteration 3: Global Deviance = 10514.72
## GAMLSS-CG iteration 4: Global Deviance = 10430.03
## GAMLSS-CG iteration 5: Global Deviance = 10418.06
## GAMLSS-CG iteration 6: Global Deviance = 10325.06
## GAMLSS-CG iteration 7: Global Deviance = 10294.18
## GAMLSS-CG iteration 8: Global Deviance = 10216.11
## GAMLSS-CG iteration 9: Global Deviance = 10209.82
## GAMLSS-CG iteration 10: Global Deviance = 10204.84
## GAMLSS-CG iteration 11: Global Deviance = 10146.2
## GAMLSS-CG iteration 12: Global Deviance = 10088.3
## GAMLSS-CG iteration 13: Global Deviance = 10030.06
## GAMLSS-CG iteration 14: Global Deviance = 9970.113
## GAMLSS-CG iteration 15: Global Deviance = 9931.734
## GAMLSS-CG iteration 16: Global Deviance = 9912.692
## GAMLSS-CG iteration 17: Global Deviance = 9874.67
## GAMLSS-CG iteration 18: Global Deviance = 9857.353
## GAMLSS-CG iteration 19: Global Deviance = 9817.291
## GAMLSS-CG iteration 20: Global Deviance = 9814.47
## GAMLSS-CG iteration 21: Global Deviance = 9764.592
## GAMLSS-CG iteration 22: Global Deviance = 9715.152
## GAMLSS-CG iteration 23: Global Deviance = 9666.486
## GAMLSS-CG iteration 24: Global Deviance = 9623.911
## GAMLSS-CG iteration 25: Global Deviance = 9590.249
## GAMLSS-CG iteration 26: Global Deviance = 9544.08
## GAMLSS-CG iteration 27: Global Deviance = 9504.707
## GAMLSS-CG iteration 28: Global Deviance = 9473.718
## GAMLSS-CG iteration 29: Global Deviance = 9428.565
## GAMLSS-CG iteration 30: Global Deviance = 9387.475
## GAMLSS-CG iteration 31: Global Deviance = 9349.301
## GAMLSS-CG iteration 32: Global Deviance = 9311.437
## GAMLSS-CG iteration 33: Global Deviance = 9275.438
## GAMLSS-CG iteration 34: Global Deviance = 9235.921
## GAMLSS-CG iteration 35: Global Deviance = 9198.974
## GAMLSS-CG iteration 36: Global Deviance = 9161.717
## GAMLSS-CG iteration 37: Global Deviance = 9126.487
## GAMLSS-CG iteration 38: Global Deviance = 9087.699
## GAMLSS-CG iteration 39: Global Deviance = 9051.199
## GAMLSS-CG iteration 40: Global Deviance = 9014.833
## GAMLSS-CG iteration 41: Global Deviance = 8979.081
## GAMLSS-CG iteration 42: Global Deviance = 8942.637
## GAMLSS-CG iteration 43: Global Deviance = 8907.442
## GAMLSS-CG iteration 44: Global Deviance = 8870.94
## GAMLSS-CG iteration 45: Global Deviance = 8835.707
## GAMLSS-CG iteration 46: Global Deviance = 8799.978
## GAMLSS-CG iteration 47: Global Deviance = 8765.085
## GAMLSS-CG iteration 48: Global Deviance = 8729.592
## GAMLSS-CG iteration 49: Global Deviance = 8694.897
## GAMLSS-CG iteration 50: Global Deviance = 8659.816
```

```
## Warning in CG(50): Algorithm CG has not yet converged
```

```
m1.cg
```

```
##
## Family:  c("SEP4", "skew exponential power type 4")
## Fitting method: CG(50)
##
## Call:  gamlss(formula = OILPRICE ~ ., family = SEP4, data = train,
##     method = CG(50))
##
## Mu Coefficients:
## (Intercept)      CL2_log      CL3_log      CL4_log      CL5_log      CL6_log
##    2.346604    -7.335423    16.620866   -10.595868    -4.518412     4.702428
##     CL7_log      CL8_log      CL9_log     CL10_log     CL11_log     CL12_log
##    3.995346    -9.606404    16.310166   -18.516306    17.068431    -9.379599
##    CL13_log     CL14_log     CL15_log     BDIY_log      SPX_log      DX1_log
##   -3.503891    -0.564837     4.696610    -0.005352    -0.298395    -0.374358
##     GC1_log      HO1_log     USCI_log      GNR_log   SHCOMP_log     FTSE_log
##   -0.148607    -0.032915    -0.012429     0.177491    -0.019310     0.312988
##     respLAG
##    1.505860
## Sigma Coefficients:
## (Intercept)
##       -3.08
## Nu Coefficients:
## (Intercept)
##       2.147
## Tau Coefficients:
## (Intercept)
##       1.235
##
##  Degrees of Freedom for the fit: 28 Residual Deg. of Freedom   772
## Global Deviance:     8659.82
##             AIC:     8715.82
##             SBC:     8846.99
```

Because the model did not converge using the CG algorithm, we can try to fit the model with the RS or mixed algorithms

```
m1.rs =gamlss(OILPRICE ~., family = SHASH,
      data = train, method=RS(50))
```

```
## GAMLSS-RS iteration 1: Global Deviance = -4338.084
## GAMLSS-RS iteration 2: Global Deviance = -4358.046
## GAMLSS-RS iteration 3: Global Deviance = -4359.165
## GAMLSS-RS iteration 4: Global Deviance = -4359.95
## GAMLSS-RS iteration 5: Global Deviance = -4360.51
## GAMLSS-RS iteration 6: Global Deviance = -4360.911
## GAMLSS-RS iteration 7: Global Deviance = -4361.2
## GAMLSS-RS iteration 8: Global Deviance = -4361.409
## GAMLSS-RS iteration 9: Global Deviance = -4361.56
## GAMLSS-RS iteration 10: Global Deviance = -4361.669
## GAMLSS-RS iteration 11: Global Deviance = -4361.749
## GAMLSS-RS iteration 12: Global Deviance = -4361.809
## GAMLSS-RS iteration 13: Global Deviance = -4361.851
## GAMLSS-RS iteration 14: Global Deviance = -4361.883
## GAMLSS-RS iteration 15: Global Deviance = -4361.907
## GAMLSS-RS iteration 16: Global Deviance = -4361.925
## GAMLSS-RS iteration 17: Global Deviance = -4361.938
## GAMLSS-RS iteration 18: Global Deviance = -4361.948
## GAMLSS-RS iteration 19: Global Deviance = -4361.956
## GAMLSS-RS iteration 20: Global Deviance = -4361.961
## GAMLSS-RS iteration 21: Global Deviance = -4361.965
## GAMLSS-RS iteration 22: Global Deviance = -4361.968
## GAMLSS-RS iteration 23: Global Deviance = -4361.971
## GAMLSS-RS iteration 24: Global Deviance = -4361.973
## GAMLSS-RS iteration 25: Global Deviance = -4361.975
## GAMLSS-RS iteration 26: Global Deviance = -4361.976
## GAMLSS-RS iteration 27: Global Deviance = -4361.977
## GAMLSS-RS iteration 28: Global Deviance = -4361.978
```

```
m1.rs
```

```
##
## Family:  c("SHASH", "Sinh-Arcsinh")
## Fitting method: RS(50)
##
## Call:  gamlss(formula = OILPRICE ~ ., family = SHASH, data = train,
##     method = RS(50))
##
## Mu Coefficients:
## (Intercept)      CL2_log      CL3_log      CL4_log      CL5_log      CL6_log
##    0.569714    -2.075595     5.541935    -3.370397    -5.781030     9.354859
##     CL7_log      CL8_log      CL9_log     CL10_log     CL11_log     CL12_log
##   -8.710302     9.498089    -6.250139     2.750065    -2.053440     1.010545
##    CL13_log     CL14_log     CL15_log     BDIY_log      SPX_log      DX1_log
##    0.327805    -0.133093    -0.311528    -0.006581    -0.061434    -0.058568
##     GC1_log      HO1_log     USCI_log      GNR_log   SHCOMP_log     FTSE_log
##   -0.028842    -0.013462    -0.012862    -0.038648    -0.028054     0.098648
##     respLAG
##    1.195275
## Sigma Coefficients:
## (Intercept)
##      -5.107
## Nu Coefficients:
## (Intercept)
##     -0.5959
## Tau Coefficients:
## (Intercept)
##      -0.533
##
##  Degrees of Freedom for the fit: 28 Residual Deg. of Freedom   772
## Global Deviance:     -4361.98
##             AIC:     -4305.98
##             SBC:     -4174.81
```

We can see above that the model has converged in 28 iterations. Hence we can try mixed algorithms to see if we can improve convergence

```
m1.mixed =gamlss(OILPRICE ~., family = SHASH,
      data = train, method=mixed(50))
```

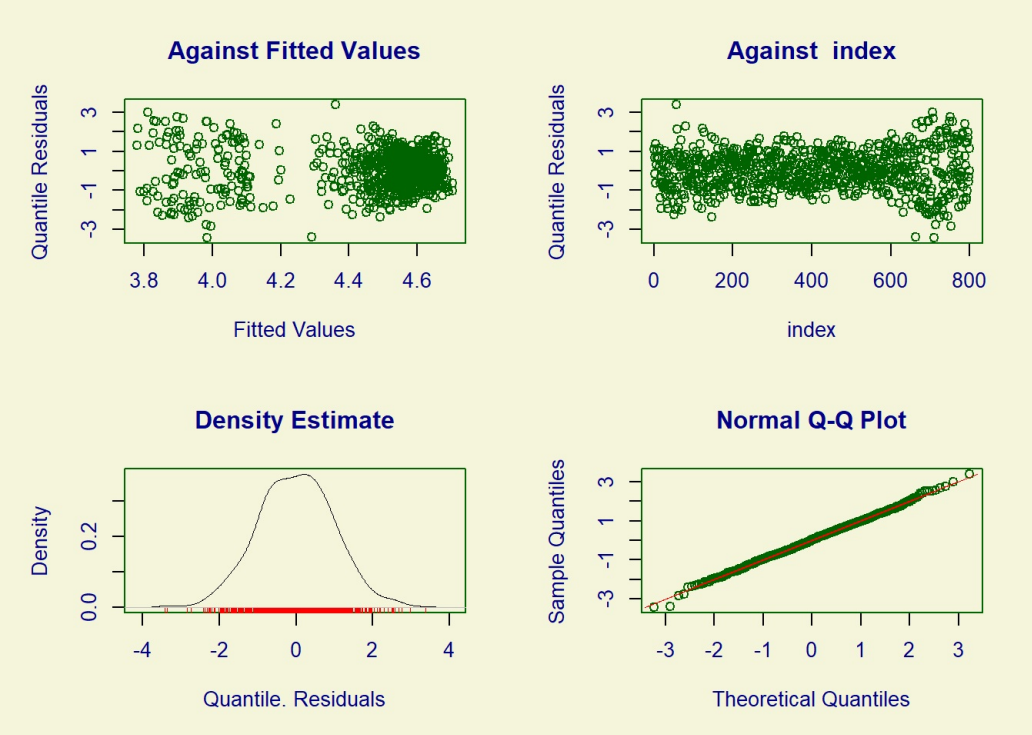
```
## GAMLSS-RS iteration 1: Global Deviance = -4338.084
## GAMLSS-RS iteration 2: Global Deviance = -4358.046
## GAMLSS-RS iteration 3: Global Deviance = -4359.165
## GAMLSS-RS iteration 4: Global Deviance = -4359.95
## GAMLSS-RS iteration 5: Global Deviance = -4360.51
## GAMLSS-RS iteration 6: Global Deviance = -4360.911
## GAMLSS-RS iteration 7: Global Deviance = -4361.2
## GAMLSS-RS iteration 8: Global Deviance = -4361.409
## GAMLSS-RS iteration 9: Global Deviance = -4361.56
## GAMLSS-RS iteration 10: Global Deviance = -4361.669
## GAMLSS-RS iteration 11: Global Deviance = -4361.749
## GAMLSS-RS iteration 12: Global Deviance = -4361.809
## GAMLSS-RS iteration 13: Global Deviance = -4361.851
## GAMLSS-RS iteration 14: Global Deviance = -4361.883
## GAMLSS-RS iteration 15: Global Deviance = -4361.907
## GAMLSS-RS iteration 16: Global Deviance = -4361.925
## GAMLSS-RS iteration 17: Global Deviance = -4361.938
## GAMLSS-RS iteration 18: Global Deviance = -4361.948
## GAMLSS-RS iteration 19: Global Deviance = -4361.956
## GAMLSS-RS iteration 20: Global Deviance = -4361.961
## GAMLSS-RS iteration 21: Global Deviance = -4361.965
## GAMLSS-RS iteration 22: Global Deviance = -4361.968
## GAMLSS-RS iteration 23: Global Deviance = -4361.971
## GAMLSS-RS iteration 24: Global Deviance = -4361.973
## GAMLSS-RS iteration 25: Global Deviance = -4361.975
## GAMLSS-RS iteration 26: Global Deviance = -4361.976
## GAMLSS-RS iteration 27: Global Deviance = -4361.977
## GAMLSS-RS iteration 28: Global Deviance = -4361.978
## GAMLSS-CG iteration 1: Global Deviance = -4361.979
```

```
m1.mixed
```

```
##
## Family:  c("SHASH", "Sinh-Arcsinh")
## Fitting method: mixed(50)
##
## Call:  gamlss(formula = OILPRICE ~ ., family = SHASH, data = train,
##     method = mixed(50))
##
## Mu Coefficients:
## (Intercept)      CL2_log      CL3_log      CL4_log      CL5_log      CL6_log
##    0.569424    -2.076356     5.541664    -3.367288    -5.785354     9.357097
##     CL7_log      CL8_log      CL9_log     CL10_log     CL11_log     CL12_log
##   -8.710585     9.498550    -6.252851     2.751946    -2.052567     1.011901
##    CL13_log     CL14_log     CL15_log     BDIY_log      SPX_log      DX1_log
##    0.324889    -0.131897    -0.311693    -0.006581    -0.061445    -0.058653
##     GC1_log      HO1_log     USCI_log      GNR_log   SHCOMP_log     FTSE_log
##   -0.028825    -0.013488    -0.012839    -0.038718    -0.028054     0.098743
##     respLAG
##    1.195593
## Sigma Coefficients:
## (Intercept)
##      -5.107
## Nu Coefficients:
## (Intercept)
##     -0.5958
## Tau Coefficients:
## (Intercept)
##     -0.5326
##
##  Degrees of Freedom for the fit: 28 Residual Deg. of Freedom   772
## Global Deviance:     -4361.98
##             AIC:     -4305.98
##             SBC:     -4174.81
```

Both RS and mixed seem to show similar convergence patterns , hence we can plot the worm plots to get a better visual understanding

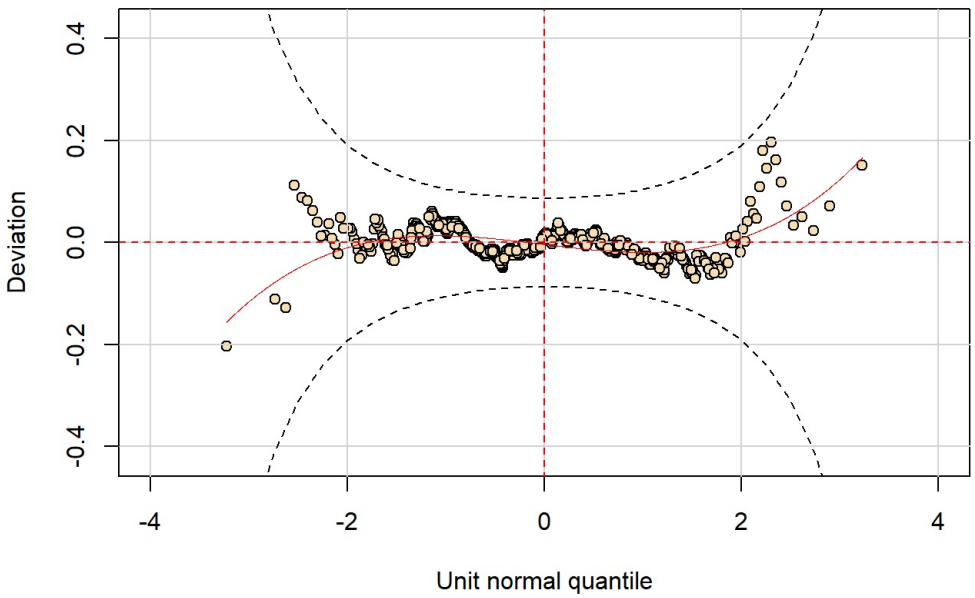
```
# RS Type
plot(m1.rs)
```



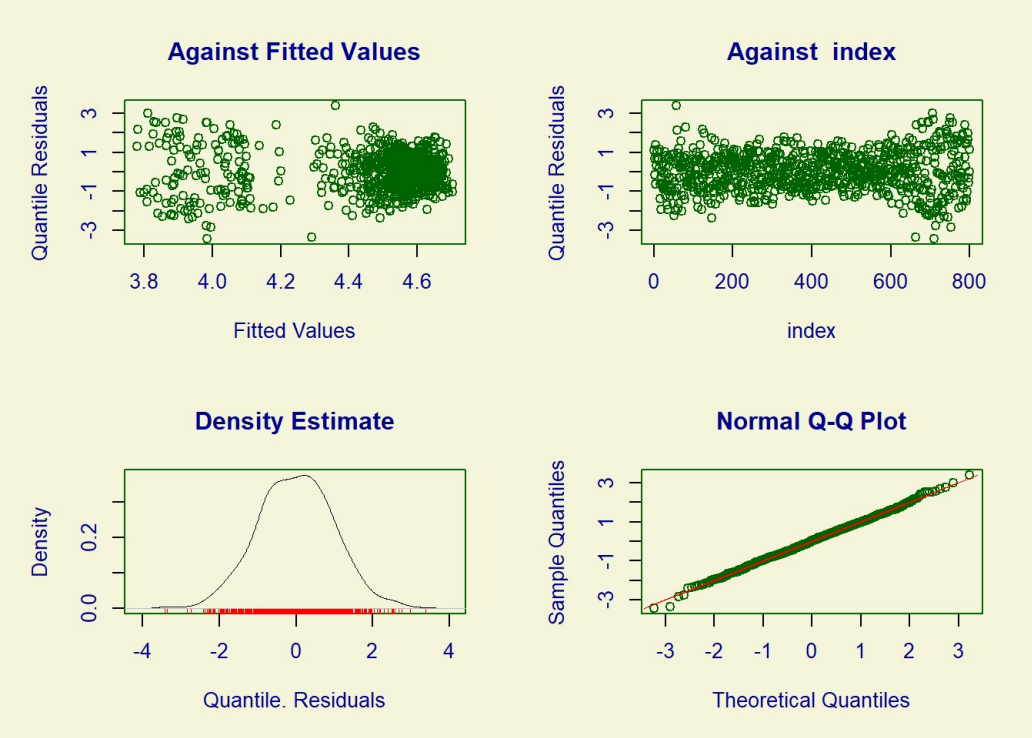

```
## ******************************************************************
##        Summary of the Quantile Residuals
##                             mean   =  -0.002830369
##                         variance   =  0.9958051
##                 coef. of skewness  =  0.003303782
##                 coef. of kurtosis  =  3.140381
## Filliben correlation coefficient  =  0.9994101
## ******************************************************************
```

```
wp(m1.rs)
```

```
## Warning in wp(m1.rs): Some points are missed out
## increase the y limits using ylim.all
```



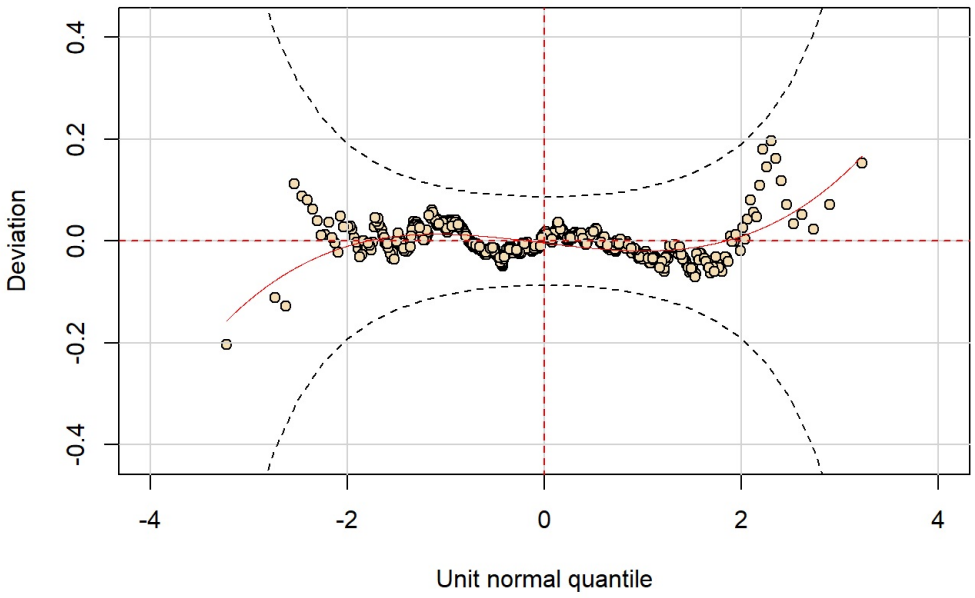

```
# Mixed Type
plot(m1.mixed)
```




```
## ****************************************************************
##        Summary of the Quantile Residuals
##                             mean   =  -0.002786998
##                         variance   =  0.995871
##                 coef. of skewness  =  0.00339154
##                 coef. of kurtosis  =  3.141029
## Filliben correlation coefficient  =  0.9994091
## ****************************************************************
```

```
wp(m1.mixed)
```

```
## Warning in wp(m1.mixed): Some points are missed out
## increase the y limits using ylim.all
```



Since the mixed gives slighly better results we use the mixed algorithm.

Using Additive terms

While trying different functions, the penalized beta splines family seems to be the most approprite function to try.

Additive terms	R function names
boosting	`boost()`
cubic splines based	`cs()`, `scs()`, `vc()`
decision trees	`tr()`
fractional and power polynomials	`fp()`, `pp()`
free knot smoothing (break points)	`fk()`
`loess`	`lo()`
neural networks	`nn()`
non-linear fit	`nl()`
penalized Beta splines based	`pb()`, `ps()`, `cy()`, `tp()`, `pvc()`
random effects	`random()` `ra()`, `rc()`, `re()`
ridge regression	`ri()`, `ridge()`
Simon Wood's gam	`ga()`

Table 2.4: Additive terms implemented within the **gamlss** packages

```
m1.add =gamlss(OILPRICE ~  cy(BDIY_log) + cy(SPX_log) + cy(DX1_log) +cy(GC1_log) +
      cy(USCI_log) + cy(GNR_log) + cy(SHCOMP_log) + cy(FTSE_log) + cy(respLAG), family = SHASH,
      data = train, method=mixed(50))
```

```
## GAMLSS-RS iteration 1: Global Deviance = -3490.322
## GAMLSS-RS iteration 2: Global Deviance = -3984.134
## GAMLSS-RS iteration 3: Global Deviance = -4082.609
## GAMLSS-RS iteration 4: Global Deviance = -4135.476
## GAMLSS-RS iteration 5: Global Deviance = -4165.009
## GAMLSS-RS iteration 6: Global Deviance = -4183.514
## GAMLSS-RS iteration 7: Global Deviance = -4207.734
## GAMLSS-RS iteration 8: Global Deviance = -4222.361
## GAMLSS-RS iteration 9: Global Deviance = -4233.265
## GAMLSS-RS iteration 10: Global Deviance = -4244.601
## GAMLSS-RS iteration 11: Global Deviance = -4253.253
## GAMLSS-RS iteration 12: Global Deviance = -4263.086
## GAMLSS-RS iteration 13: Global Deviance = -4269.373
## GAMLSS-RS iteration 14: Global Deviance = -4273.823
## GAMLSS-RS iteration 15: Global Deviance = -4278.028
## GAMLSS-RS iteration 16: Global Deviance = -4281.238
## GAMLSS-RS iteration 17: Global Deviance = -4284.352
## GAMLSS-RS iteration 18: Global Deviance = -4286.879
## GAMLSS-RS iteration 19: Global Deviance = -4288.787
## GAMLSS-RS iteration 20: Global Deviance = -4291.213
## GAMLSS-RS iteration 21: Global Deviance = -4292.892
## GAMLSS-RS iteration 22: Global Deviance = -4294.783
## GAMLSS-RS iteration 23: Global Deviance = -4296.401
## GAMLSS-RS iteration 24: Global Deviance = -4297.623
## GAMLSS-RS iteration 25: Global Deviance = -4297.807
## GAMLSS-RS iteration 26: Global Deviance = -4299.195
## GAMLSS-RS iteration 27: Global Deviance = -4300.111
## GAMLSS-RS iteration 28: Global Deviance = -4300.682
## GAMLSS-RS iteration 29: Global Deviance = -4300.966
## GAMLSS-RS iteration 30: Global Deviance = -4301.546
## GAMLSS-RS iteration 31: Global Deviance = -4302.301
## GAMLSS-RS iteration 32: Global Deviance = -4303.081
## GAMLSS-RS iteration 33: Global Deviance = -4303.615
## GAMLSS-RS iteration 34: Global Deviance = -4303.037
## GAMLSS-RS iteration 35: Global Deviance = -4302.711
## GAMLSS-RS iteration 36: Global Deviance = -4302.249
## GAMLSS-RS iteration 37: Global Deviance = -4301.047
## GAMLSS-RS iteration 38: Global Deviance = -4303.574
## GAMLSS-RS iteration 39: Global Deviance = -4304.742
## GAMLSS-RS iteration 40: Global Deviance = -4304.085
## GAMLSS-RS iteration 41: Global Deviance = -4305.027
## GAMLSS-RS iteration 42: Global Deviance = -4305.811
## GAMLSS-RS iteration 43: Global Deviance = -4306.088
## GAMLSS-RS iteration 44: Global Deviance = -4305.878
## GAMLSS-RS iteration 45: Global Deviance = -4303.975
## GAMLSS-RS iteration 46: Global Deviance = -4304.444
## GAMLSS-RS iteration 47: Global Deviance = -4305.834
## GAMLSS-RS iteration 48: Global Deviance = -4307.319
## GAMLSS-RS iteration 49: Global Deviance = -4308.213
## GAMLSS-RS iteration 50: Global Deviance = -4309.159
## GAMLSS-CG iteration 1: Global Deviance = -3524.155
## GAMLSS-CG iteration 2: Global Deviance = -3723.72
## GAMLSS-CG iteration 3: Global Deviance = -3888.081
## GAMLSS-CG iteration 4: Global Deviance = -3940.162
## GAMLSS-CG iteration 5: Global Deviance = -4002.743
## GAMLSS-CG iteration 6: Global Deviance = -4039.941
## GAMLSS-CG iteration 7: Global Deviance = -4079.256
## GAMLSS-CG iteration 8: Global Deviance = -4089.363
## GAMLSS-CG iteration 9: Global Deviance = -4129.738
## GAMLSS-CG iteration 10: Global Deviance = -4137.466
## GAMLSS-CG iteration 11: Global Deviance = -4168.182
## GAMLSS-CG iteration 12: Global Deviance = -4178.388
## GAMLSS-CG iteration 13: Global Deviance = -4191.78
## GAMLSS-CG iteration 14: Global Deviance = -4199.461
## GAMLSS-CG iteration 15: Global Deviance = -4210.415
## GAMLSS-CG iteration 16: Global Deviance = -4213.38
## GAMLSS-CG iteration 17: Global Deviance = -4213.841
## GAMLSS-CG iteration 18: Global Deviance = -4233.539
## GAMLSS-CG iteration 19: Global Deviance = -4258.691
## GAMLSS-CG iteration 20: Global Deviance = -4283.055
```

```
## Warning in CG(n.cyc = n2): Algorithm CG has not yet converged
```

```
m1.add
```

```
##
## Family:  c("SHASH", "Sinh-Arcsinh")
## Fitting method: mixed(50)
##
## Call:  gamlss(formula = OILPRICE ~ cy(BDIY_log) + cy(SPX_log) +
##     cy(DX1_log) + cy(GC1_log) + cy(USCI_log) + cy(GNR_log) +
##     cy(SHCOMP_log) + cy(FTSE_log) + cy(respLAG), family = SHASH,
##     data = train, method = mixed(50))
##
## Mu Coefficients:
##    (Intercept)    cy(BDIY_log)     cy(SPX_log)     cy(DX1_log)     cy(GC1_log)
##          4.529              NA              NA              NA              NA
##   cy(USCI_log)     cy(GNR_log)  cy(SHCOMP_log)    cy(FTSE_log)     cy(respLAG)
##             NA              NA              NA              NA              NA
## Sigma Coefficients:
## (Intercept)
##      -5.793
## Nu Coefficients:
## (Intercept)
##     -0.8894
## Tau Coefficients:
## (Intercept)
##     -0.9242
##
##  Degrees of Freedom for the fit: 110.9 Residual Deg. of Freedom   689.1
## Global Deviance:     -4283.05
##             AIC:     -4061.3
##             SBC:     -3541.9
```

We try another spline function from the same family.

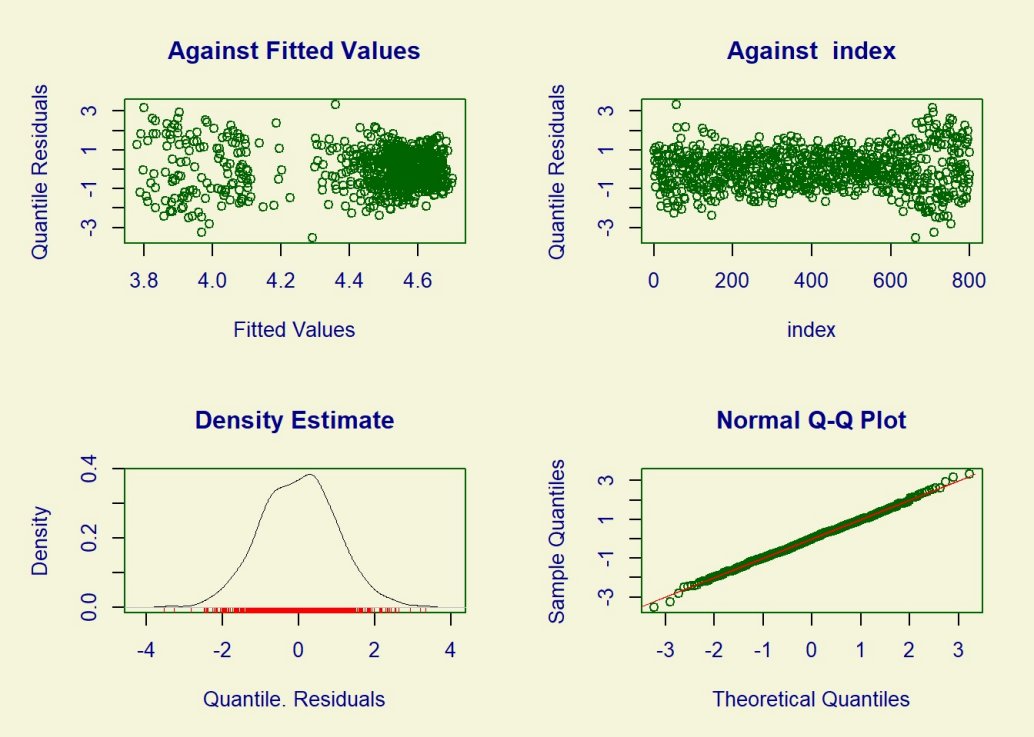
```
m2.add =gamlss(OILPRICE ~  pb(BDIY_log) + pb(SPX_log) + pb(DX1_log) + pb(GC1_log) +
      pb(USCI_log) + pb(GNR_log) + pb(SHCOMP_log) + pb(FTSE_log)+ pb(respLAG), family = SHASH,
      data = train, method=mixed(50))
```

```
## GAMLSS-RS iteration 1: Global Deviance = -4333.936
## GAMLSS-RS iteration 2: Global Deviance = -4354.952
## GAMLSS-RS iteration 3: Global Deviance = -4355.968
## GAMLSS-RS iteration 4: Global Deviance = -4356.66
## GAMLSS-RS iteration 5: Global Deviance = -4357.066
## GAMLSS-RS iteration 6: Global Deviance = -4357.305
## GAMLSS-RS iteration 7: Global Deviance = -4357.441
## GAMLSS-RS iteration 8: Global Deviance = -4357.514
## GAMLSS-RS iteration 9: Global Deviance = -4357.547
## GAMLSS-RS iteration 10: Global Deviance = -4357.556
## GAMLSS-RS iteration 11: Global Deviance = -4357.546
## GAMLSS-RS iteration 12: Global Deviance = -4357.523
## GAMLSS-RS iteration 13: Global Deviance = -4357.493
## GAMLSS-RS iteration 14: Global Deviance = -4357.46
## GAMLSS-RS iteration 15: Global Deviance = -4357.427
## GAMLSS-RS iteration 16: Global Deviance = -4357.395
## GAMLSS-RS iteration 17: Global Deviance = -4357.364
## GAMLSS-RS iteration 18: Global Deviance = -4357.336
## GAMLSS-RS iteration 19: Global Deviance = -4357.31
## GAMLSS-RS iteration 20: Global Deviance = -4357.286
## GAMLSS-RS iteration 21: Global Deviance = -4357.265
## GAMLSS-RS iteration 22: Global Deviance = -4357.245
## GAMLSS-RS iteration 23: Global Deviance = -4357.229
## GAMLSS-RS iteration 24: Global Deviance = -4357.214
## GAMLSS-RS iteration 25: Global Deviance = -4357.201
## GAMLSS-RS iteration 26: Global Deviance = -4357.19
## GAMLSS-RS iteration 27: Global Deviance = -4357.18
## GAMLSS-RS iteration 28: Global Deviance = -4357.17
## GAMLSS-RS iteration 29: Global Deviance = -4357.162
## GAMLSS-RS iteration 30: Global Deviance = -4357.155
## GAMLSS-RS iteration 31: Global Deviance = -4357.148
## GAMLSS-RS iteration 32: Global Deviance = -4357.142
## GAMLSS-RS iteration 33: Global Deviance = -4357.137
## GAMLSS-RS iteration 34: Global Deviance = -4357.132
## GAMLSS-RS iteration 35: Global Deviance = -4357.128
## GAMLSS-RS iteration 36: Global Deviance = -4357.124
## GAMLSS-RS iteration 37: Global Deviance = -4357.121
## GAMLSS-RS iteration 38: Global Deviance = -4357.118
## GAMLSS-RS iteration 39: Global Deviance = -4357.115
## GAMLSS-RS iteration 40: Global Deviance = -4357.112
## GAMLSS-RS iteration 41: Global Deviance = -4357.11
## GAMLSS-RS iteration 42: Global Deviance = -4357.108
## GAMLSS-RS iteration 43: Global Deviance = -4357.106
## GAMLSS-RS iteration 44: Global Deviance = -4357.105
## GAMLSS-RS iteration 45: Global Deviance = -4357.103
## GAMLSS-RS iteration 46: Global Deviance = -4357.102
## GAMLSS-RS iteration 47: Global Deviance = -4357.101
## GAMLSS-RS iteration 48: Global Deviance = -4357.1
## GAMLSS-CG iteration 1: Global Deviance = -4357.095
## GAMLSS-CG iteration 2: Global Deviance = -4357.094
```

```
m2.add
```

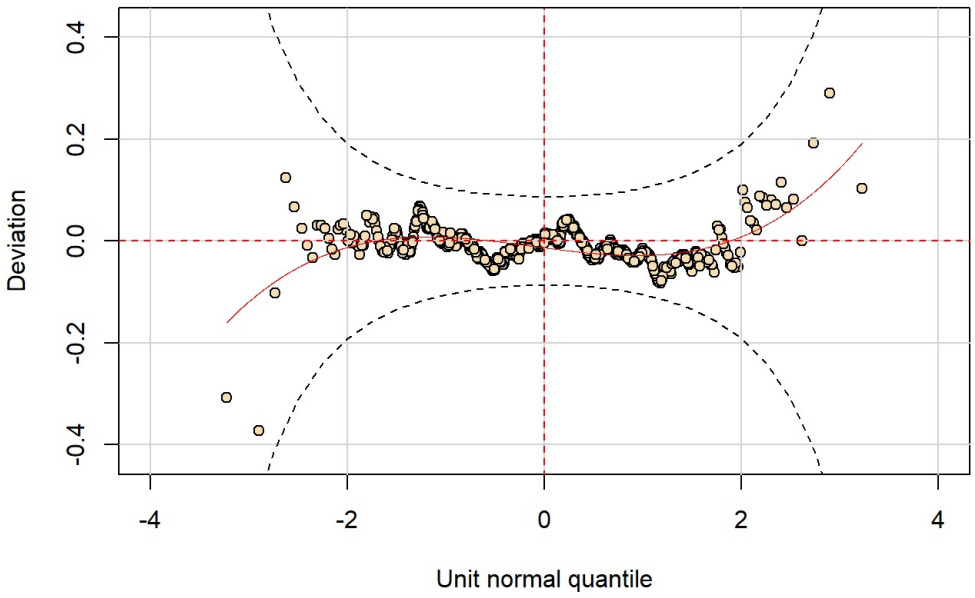
```
##
## Family:  c("SHASH", "Sinh-Arcsinh")
## Fitting method: mixed(50)
##
## Call:  gamlss(formula = OILPRICE ~ pb(BDIY_log) + pb(SPX_log) +
##     pb(DX1_log) + pb(GC1_log) + pb(USCI_log) + pb(GNR_log) +
##     pb(SHCOMP_log) + pb(FTSE_log) + pb(respLAG), family = SHASH,
##     data = train, method = mixed(50))
##
## Mu Coefficients:
##    (Intercept)    pb(BDIY_log)     pb(SPX_log)     pb(DX1_log)     pb(GC1_log)
##       0.463296       -0.004159       -0.061650        0.015801       -0.051762
##   pb(USCI_log)     pb(GNR_log)  pb(SHCOMP_log)    pb(FTSE_log)     pb(respLAG)
##       0.024287        0.028061       -0.014907        0.038196        0.978257
## Sigma Coefficients:
## (Intercept)
##      -5.084
## Nu Coefficients:
## (Intercept)
##     -0.5685
## Tau Coefficients:
## (Intercept)
##     -0.5404
##
##  Degrees of Freedom for the fit: 26.09 Residual Deg. of Freedom   773.9
## Global Deviance:     -4357.09
##             AIC:     -4304.91
##             SBC:     -4182.69
```

```
# Plot additive model with pb spline
plot(m2.add)
```




```
## ******************************************************************
##        Summary of the Quantile Residuals
##                             mean   =  -0.01032291
##                         variance   =  0.9954453
##                 coef. of skewness  =  0.01619591
##                 coef. of kurtosis  =  3.155239
## Filliben correlation coefficient  =  0.9993526
## ******************************************************************
```

```
wp(m2.add)
```



This is not significantly better than our previous model . Let us try some more

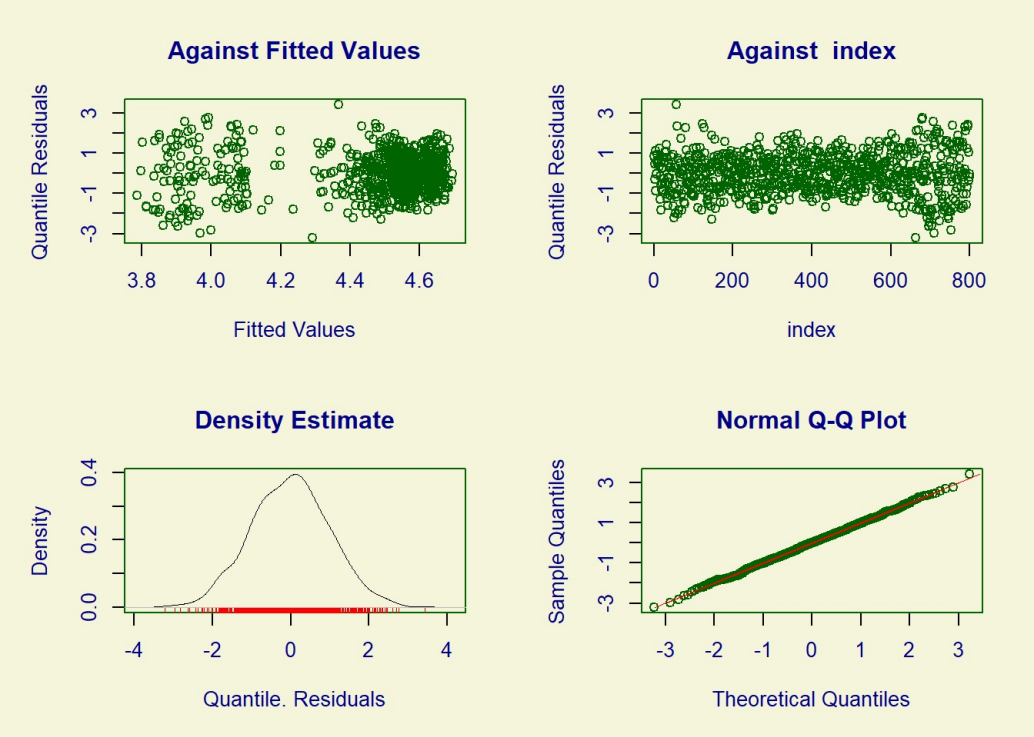
```
m3.add =gamlss(OILPRICE ~ ps(BDIY_log) + ps(SPX_log) + ps(DX1_log) + ps(GC1_log) +
      ps(USCI_log) + ps(GNR_log) + ps(SHCOMP_log) + ps(FTSE_log)+ ps(respLAG) , family = SHASH,
      data = train, method=mixed(50))
```

```
## GAMLSS-RS iteration 1: Global Deviance = -4416.858
## GAMLSS-RS iteration 2: Global Deviance = -4504.806
## GAMLSS-RS iteration 3: Global Deviance = -4506.601
## GAMLSS-RS iteration 4: Global Deviance = -4506.684
## GAMLSS-RS iteration 5: Global Deviance = -4506.713
## GAMLSS-RS iteration 6: Global Deviance = -4506.723
## GAMLSS-RS iteration 7: Global Deviance = -4506.729
## GAMLSS-RS iteration 8: Global Deviance = -4506.734
## GAMLSS-RS iteration 9: Global Deviance = -4506.742
## GAMLSS-RS iteration 10: Global Deviance = -4506.752
## GAMLSS-RS iteration 11: Global Deviance = -4506.764
## GAMLSS-RS iteration 12: Global Deviance = -4506.777
## GAMLSS-RS iteration 13: Global Deviance = -4506.793
## GAMLSS-RS iteration 14: Global Deviance = -4506.812
## GAMLSS-RS iteration 15: Global Deviance = -4506.832
## GAMLSS-RS iteration 16: Global Deviance = -4506.854
## GAMLSS-RS iteration 17: Global Deviance = -4506.877
## GAMLSS-RS iteration 18: Global Deviance = -4506.902
## GAMLSS-RS iteration 19: Global Deviance = -4506.927
## GAMLSS-RS iteration 20: Global Deviance = -4506.952
## GAMLSS-RS iteration 21: Global Deviance = -4506.976
## GAMLSS-RS iteration 22: Global Deviance = -4507.001
## GAMLSS-RS iteration 23: Global Deviance = -4507.025
## GAMLSS-RS iteration 24: Global Deviance = -4507.048
## GAMLSS-RS iteration 25: Global Deviance = -4507.071
## GAMLSS-RS iteration 26: Global Deviance = -4507.092
## GAMLSS-RS iteration 27: Global Deviance = -4507.114
## GAMLSS-RS iteration 28: Global Deviance = -4507.136
## GAMLSS-RS iteration 29: Global Deviance = -4507.156
## GAMLSS-RS iteration 30: Global Deviance = -4507.175
## GAMLSS-RS iteration 31: Global Deviance = -4507.194
## GAMLSS-RS iteration 32: Global Deviance = -4507.212
## GAMLSS-RS iteration 33: Global Deviance = -4507.231
## GAMLSS-RS iteration 34: Global Deviance = -4507.249
## GAMLSS-RS iteration 35: Global Deviance = -4507.266
## GAMLSS-RS iteration 36: Global Deviance = -4507.282
## GAMLSS-RS iteration 37: Global Deviance = -4507.297
## GAMLSS-RS iteration 38: Global Deviance = -4507.314
## GAMLSS-RS iteration 39: Global Deviance = -4507.329
## GAMLSS-RS iteration 40: Global Deviance = -4507.344
## GAMLSS-RS iteration 41: Global Deviance = -4507.358
## GAMLSS-RS iteration 42: Global Deviance = -4507.372
## GAMLSS-RS iteration 43: Global Deviance = -4507.386
## GAMLSS-RS iteration 44: Global Deviance = -4507.399
## GAMLSS-RS iteration 45: Global Deviance = -4507.411
## GAMLSS-RS iteration 46: Global Deviance = -4507.425
## GAMLSS-RS iteration 47: Global Deviance = -4507.438
## GAMLSS-RS iteration 48: Global Deviance = -4507.449
## GAMLSS-RS iteration 49: Global Deviance = -4507.461
## GAMLSS-RS iteration 50: Global Deviance = -4507.471
## GAMLSS-CG iteration 1: Global Deviance = -4498.714
## GAMLSS-CG iteration 2: Global Deviance = -4502.551
## GAMLSS-CG iteration 3: Global Deviance = -4505.024
## GAMLSS-CG iteration 4: Global Deviance = -4505.079
## GAMLSS-CG iteration 5: Global Deviance = -4506.076
## GAMLSS-CG iteration 6: Global Deviance = -4506.741
## GAMLSS-CG iteration 7: Global Deviance = -4506.853
## GAMLSS-CG iteration 8: Global Deviance = -4507.155
## GAMLSS-CG iteration 9: Global Deviance = -4507.184
## GAMLSS-CG iteration 10: Global Deviance = -4507.336
## GAMLSS-CG iteration 11: Global Deviance = -4507.387
## GAMLSS-CG iteration 12: Global Deviance = -4507.465
## GAMLSS-CG iteration 13: Global Deviance = -4507.582
## GAMLSS-CG iteration 14: Global Deviance = -4507.605
## GAMLSS-CG iteration 15: Global Deviance = -4507.614
## GAMLSS-CG iteration 16: Global Deviance = -4507.683
## GAMLSS-CG iteration 17: Global Deviance = -4507.69
## GAMLSS-CG iteration 18: Global Deviance = -4507.745
## GAMLSS-CG iteration 19: Global Deviance = -4507.782
## GAMLSS-CG iteration 20: Global Deviance = -4507.782
```

```
m3.add
```

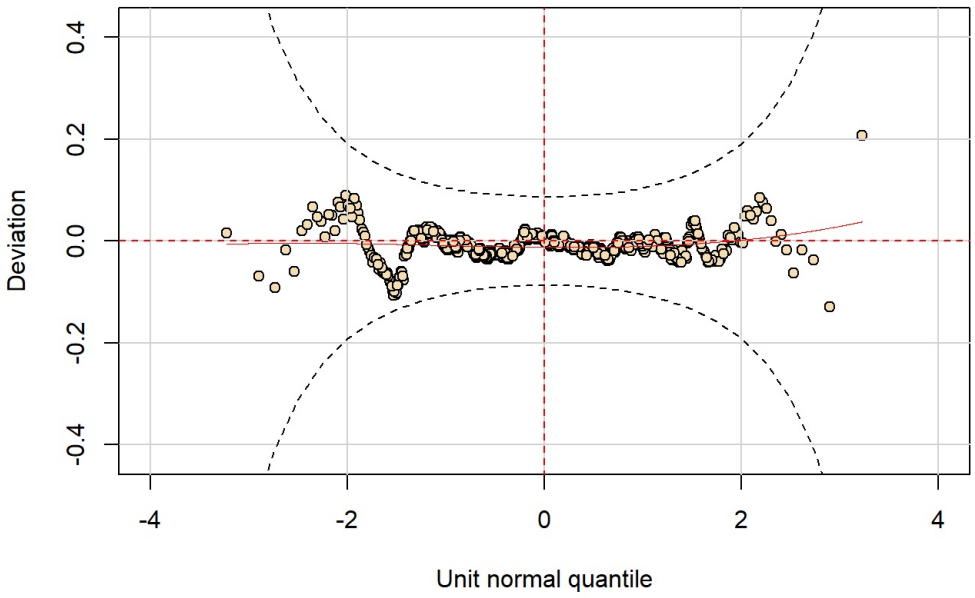
```
##
## Family:  c("SHASH", "Sinh-Arcsinh")
## Fitting method: mixed(50)
##
## Call:  gamlss(formula = OILPRICE ~ ps(BDIY_log) + ps(SPX_log) +
##     ps(DX1_log) + ps(GC1_log) + ps(USCI_log) + ps(GNR_log) +
##     ps(SHCOMP_log) + ps(FTSE_log) + ps(respLAG), family = SHASH,
##     data = train, method = mixed(50))
##
## Mu Coefficients:
##    (Intercept)    ps(BDIY_log)     ps(SPX_log)     ps(DX1_log)     ps(GC1_log)
##       3.889590       -0.006082       -0.020518       -0.334379       -0.078355
##   ps(USCI_log)     ps(GNR_log)  ps(SHCOMP_log)    ps(FTSE_log)     ps(respLAG)
##       0.019342        0.056764       -0.074164       -0.075999        0.842429
## Sigma Coefficients:
## (Intercept)
##      -5.339
## Nu Coefficients:
## (Intercept)
##     -0.6891
## Tau Coefficients:
## (Intercept)
##     -0.5704
##
##  Degrees of Freedom for the fit: 73.61 Residual Deg. of Freedom   726.4
## Global Deviance:     -4507.78
##             AIC:     -4360.56
##             SBC:     -4015.71
```

```
# Plot additive model with ps spline
plot(m3.add)
```




```
## ******************************************************************
##        Summary of the Quantile Residuals
##                             mean   =  -0.009359698
##                         variance   =  1.001861
##                 coef. of skewness  =  0.01545704
##                 coef. of kurtosis  =  2.986235
## Filliben correlation coefficient  =  0.9996876
## ******************************************************************
```

```
wp(m3.add)
```



While this model took much longer to converge, we seem to get much better results.

Comparing the Models

reference : chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.gamlss.com/wp-content/uploads/2013/01/book-2010-Athens1.pdf (extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.gamlss.com/wp-content/uploads/2013/01/book-2010-Athens1.pdf)

Different models can be compared using their global deviances, GD = −2ˆ, (if they are nested) or using a generalized Akaike information criterion, GAIC = −2ˆ+ (.df), where ˆ= n i=1 logf(yi|ˆµi, ˆσi, ˆνi, ˆ τi) is the fitted log-likelihood function and is a required penalty, e.g. =2fortheusual Akaike information criterion or = log(n) for the Schwartz Bayesian criterion. The function deviance() provides the global deviance of the model. N

Let us compare the models generated:

- Based on Algorithm

```
GAIC(m1.cg,m1.rs, m1.mixed)
```

```
##          df       AIC
## m1.mixed 28 -4305.979
## m1.rs    28 -4305.978
## m1.cg    28  8715.816
```

As expected, the mixed algorithm seems to be the most appropriate

- Based on Additive terms

```
GAIC(m1.add, m2.add, m3.add )
```

```
##               df       AIC
## m3.add  73.61318 -4360.555
## m2.add  26.09112 -4304.912
## m1.add 110.87511 -4061.305
```

```
GAIC(m1.add, m2.add, m3.add, k =3 )
```

```
##               df       AIC
## m3.add  73.61318 -4286.942
## m2.add  26.09112 -4278.821
## m1.add 110.87511 -3950.429
```

```
GAIC(m1.add, m2.add, m3.add, k =4 )
```

```
##               df       AIC
## m2.add  26.09112 -4252.730
## m3.add  73.61318 -4213.329
## m1.add 110.87511 -3839.554
```

The ps function seems to be the most appropriate. Note : The AIC function uses default penalty = 2, giving the usual Akaike information criterion (AIC). We see that k = 3 also selects 3 as the most appropriate model. However for k=4 onwards we find that m2 is more suitable. Hence, we select m2

Extracting Features of the Model

- extract the global deviance of the gamlss model object

```
deviance(m2.add)
```

```
## [1] -4357.094
```

- extract the individual effective degrees of freedom

```
edf(m2.add)
```

```
## Effective df for mu model
```

```
## $`pb(BDIY_log)`
## [1] 3.572333
##
## $`pb(SPX_log)`
## [1] 3.621015
##
## $`pb(DX1_log)`
## [1] 3.372273
##
## $`pb(GC1_log)`
## [1] 3.525722
##
## $`pb(USCI_log)`
## [1] 3.313893
##
## $`pb(GNR_log)`
## [1] 3.56085
##
## $`pb(SHCOMP_log)`
## [1] 3.325024
##
## $`pb(FTSE_log)`
## [1] 3.462878
##
## $`pb(respLAG)`
## [1] 3.337132
```

- to extract a model formula

```
formula(m2.add)
```

```
## OILPRICE ~ pb(BDIY_log) + pb(SPX_log) + pb(DX1_log) + pb(GC1_log) +
##     pb(USCI_log) + pb(GNR_log) + pb(SHCOMP_log) + pb(FTSE_log) +
##     pb(respLAG)
```

- to summarize the fit in a gamlss object

```
summary(m2.add)
```

```
## Warning in summary.gamlss(m2.add): summary: vcov has failed, option qr is used instead
```

```
## ******************************************************************
## Family:  c("SHASH", "Sinh-Arcsinh")
##
## Call:  gamlss(formula = OILPRICE ~ pb(BDIY_log) + pb(SPX_log) + pb(DX1_log) +
##     pb(GC1_log) + pb(USCI_log) + pb(GNR_log) + pb(SHCOMP_log) +
##     pb(FTSE_log) + pb(respLAG), family = SHASH, data = train,
##     method = mixed(50))
##
## Fitting method: mixed(50)
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.463296   0.282030   1.643 0.100846
## pb(BDIY_log)   -0.004159   0.002302  -1.806 0.071268 .
## pb(SPX_log)    -0.061650   0.016084  -3.833 0.000137 ***
## pb(DX1_log)     0.015801   0.034288   0.461 0.645054
## pb(GC1_log)    -0.051762   0.015765  -3.283 0.001072 **
## pb(USCI_log)    0.024287   0.026431   0.919 0.358454
## pb(GNR_log)     0.028061   0.023774   1.180 0.238231
## pb(SHCOMP_log) -0.014907   0.006053  -2.463 0.013997 *
## pb(FTSE_log)    0.038196   0.024019   1.590 0.112191
## pb(respLAG)     0.978257   0.007790 125.571  < 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) -5.08405    0.03508  -144.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Nu link function:  log
## Nu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.56855    0.02333  -24.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Tau link function:  log
## Tau Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.54037    0.02323  -23.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## NOTE: Additive smoothing terms exist in the formulas:
##  i) Std. Error for smoothers are for the linear effect only.
## ii) Std. Error for the linear terms may not be reliable.
## ------------------------------------------------------------------
## No. of observations in the fit:  800
## Degrees of Freedom for the fit:  26.09112
##       Residual Deg. of Freedom:  773.9089
##                       at cycle:  2
##
## Global Deviance:     -4357.094
##             AIC:     -4304.912
##             SBC:     -4182.685
## ******************************************************************
```

- Find the degree of freedom for hypyer parameter tuning. (the default seems to be the best option)

```
find.hyper(m2.add, parameters=c(1,5))
```

```
## par 1 5 crit= -4304.912 with pen= 2
## par 1.1 5 crit= -4304.912 with pen= 2
## par 0.9 5 crit= -4304.912 with pen= 2
## par 1 5.1 crit= -4304.912 with pen= 2
## par 1 4.9 crit= -4304.912 with pen= 2
```

```
## $par
## [1] 1 5
##
## $value
## [1] -4304.912
##
## $counts
## function gradient
##        1        1
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: NORM OF PROJECTED GRADIENT <= PGTOL"
```

Model for Prediction

Create a column in the test data for predicted values and print the first 5 values

```
test['predictions'] = predict(m2.add,newdata=test)
```

```
## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
```

```
# Print the predicted column
head(test$predictions,5)
```

```
## [1] 4.089321 4.087391 4.094292 4.080205 4.081890
```

Model Validation

Deviations of the model from true values for different confidence intervals rounded to the 5th decimal

```
test["deviations"]  = round(test['predictions']/test['OILPRICE'],5)
```

```
test_indexed <- tibble::rowid_to_column(test, "index")
```

```
# Plot predicted Value Against True Value
ggplot(test_indexed) +

  geom_line(aes(x = index, y = predictions), size = 0.5, colour = "#FF0000") +
  geom_line(aes(x = index, y = OILPRICE), size = 0.5, colour = "green")
```


```
  theme_minimal()
```

```
## List of 93
##  $ line                        :List of 6
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                        :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                        :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                       : NULL
##  $ aspect.ratio                : NULL
##  $ axis.title                  : NULL
##  $ axis.title.x                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top            :List of 11
```

```
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom       : NULL
##  $ axis.title.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left         : NULL
##  $ axis.title.y.right        :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top           :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
```

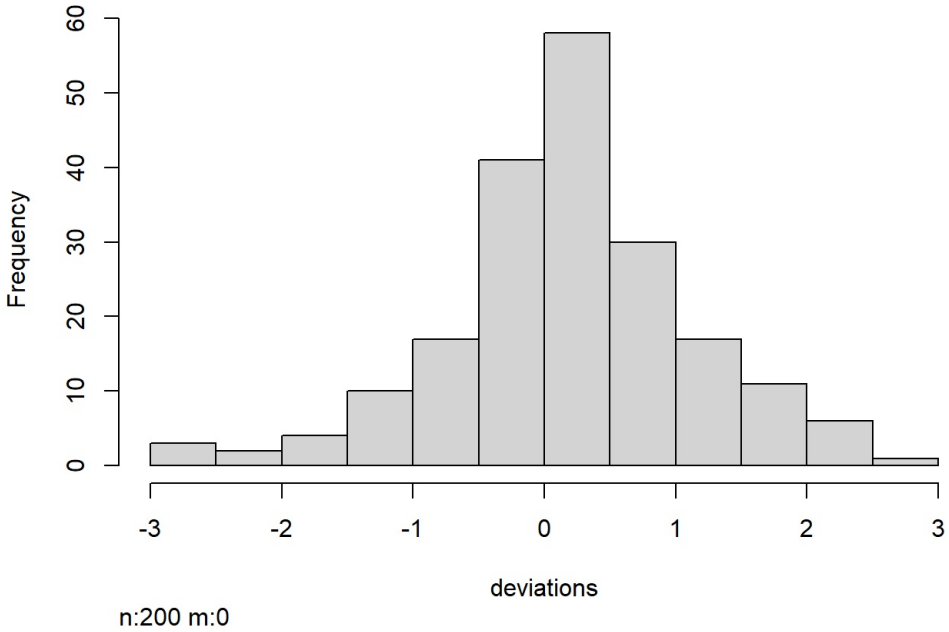
```
##   ..$ margin       : 'margin' num [1:4] 0points 0points 2.2points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.bottom        : NULL
##  $ axis.text.y               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 2.2points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left          : NULL
##  $ axis.text.y.right         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 2.2points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x              : NULL
##  $ axis.ticks.x.top          : NULL
##  $ axis.ticks.x.bottom       : NULL
##  $ axis.ticks.y              : NULL
##  $ axis.ticks.y.left         : NULL
##  $ axis.ticks.y.right        : NULL
##  $ axis.ticks.length         : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ axis.ticks.length.x       : NULL
##  $ axis.ticks.length.x.top   : NULL
##  $ axis.ticks.length.x.bottom: NULL
##  $ axis.ticks.length.y       : NULL
##  $ axis.ticks.length.y.left  : NULL
##  $ axis.ticks.length.y.right : NULL
##  $ axis.line                 : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x               : NULL
##  $ axis.line.x.top           : NULL
##  $ axis.line.x.bottom        : NULL
##  $ axis.line.y               : NULL
##  $ axis.line.y.left          : NULL
##  $ axis.line.y.right         : NULL
##  $ legend.background         : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin             : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing            : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x          : NULL
##  $ legend.spacing.y          : NULL
##  $ legend.key                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size           : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ legend.key.height         : NULL
##  $ legend.key.width          : NULL
##  $ legend.text               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
```

```
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.align         : NULL
##  $ legend.title              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.align        : NULL
##  $ legend.position           : chr "right"
##  $ legend.direction          : NULL
##  $ legend.justification      : chr "center"
##  $ legend.box                : NULL
##  $ legend.box.just           : NULL
##  $ legend.box.margin         : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "unit")= int 1
##  $ legend.box.background     : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing        : 'simpleUnit' num 11points
##   ..- attr(*, "unit")= int 8
##  $ panel.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.border              : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.spacing             : 'simpleUnit' num 5.5points
##   ..- attr(*, "unit")= int 8
##  $ panel.spacing.x           : NULL
##  $ panel.spacing.y           : NULL
##  $ panel.grid                :List of 6
##   ..$ colour       : chr "grey92"
##   ..$ size         : NULL
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major          : NULL
##  $ panel.grid.minor          :List of 6
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.5
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major.x        : NULL
##  $ panel.grid.major.y        : NULL
##  $ panel.grid.minor.x        : NULL
##  $ panel.grid.minor.y        : NULL
##  $ panel.ontop               : logi FALSE
##  $ plot.background           : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ plot.title                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
```

```
##  $ plot.title.position       : chr "panel"
##  $ plot.subtitle             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0points 0points 5.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : num 1
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 5.5points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption.position     : chr "panel"
##  $ plot.tag                  :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.tag.position         : chr "topleft"
##  $ plot.margin               : 'margin' num [1:4] 5.5points 5.5points 5.5points 5.5points
##   ..- attr(*, "unit")= int 8
##  $ strip.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ strip.background.x        : NULL
##  $ strip.background.y        : NULL
##  $ strip.placement           : chr "inside"
##  $ strip.text                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey10"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 4.4points 4.4points 4.4points 4.4points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.text.x              : NULL
##  $ strip.text.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
```

```
##  $ strip.switch.pad.grid     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.switch.pad.wrap     : 'simpleUnit' num 2.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.text.y.left         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

```
# Plot a historgram of the devaitions
hist((test["deviations"]-1)*100)
```



Most values fall below 2% of the deviation from the true values.

```
# % of values 3 % or lower
sum(test['deviations']<1.03)/nrow(test)*100
```

```
## [1] 100
```

```
# % of values 2 % or lower
sum(test['deviations']<1.02)/nrow(test)*100
```

```
## [1] 96.5
```

```
# % of values 1 % or lower
sum(test['deviations']<1.01)/nrow(test)*100
```

```
## [1] 82.5
```

```
# % of values 0.5 % or lower
sum(test['deviations']<1.005)/nrow(test)*100
```

```
## [1] 67.5
```

96.5 of the values fall below 2% of the deviations.