Time Series Homework 1

Ekaterina Petrova, Anna Zhurba, Arsenii Lishnevsky

21 02 2024

Мы выбрали для анализа и предсказаний временные ряды средневзвешенного ETC купли-продажи денежных средств в долларах США в лотах за российские рубли. Сайт источника: https://www.moex.com/a1494 (https://www.moex.com/a1494)

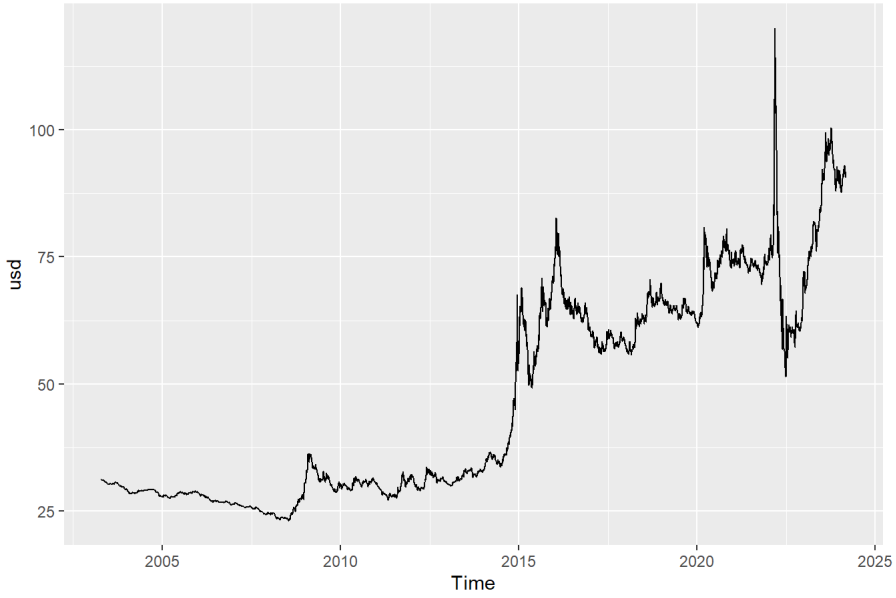
Очитска и структуризация данных

Первоначальные данные будних дней с 15.04.2003 по 05.03.2024. Первым шагом добавляем все дни в году, включающие праздничные и выходные, когда торги на бирже не велись. Дальше, удаляем все значения за 29 февраля. Следующим шагом мы предполагаем, что в дни, где предшедствующие дни пустые, участники ориентировались на предыдущие значения ставок. Поэтому заменяем пустые значения на последние известные значения дней, когда торги велись. В итоге, мы получаем временной ряд в 7625 наблюдений. Ниже можем наблюдать как распределена по времени ставка:

```
dd <- read.xlsx("USDRUB_TOM.xlsx",
                sheet = "Sheet1")
dd$Дата.торгов=as.Date(dd$Дата.торгов,origin = "1899-12-30")
dd=pad(dd)
```

```
## pad applied on the interval: day
```

```
dd$day_month = format(dd$Дата.торгов,"%m %d")
dd = dd[dd$day_month != "02 29",] # исключаем 29 февраля
dd$moex_works= ifelse(is.na(dd$Кратк..наим.), 0, 1) # даты по которым изначально не было данных (NA) - предполагаем, что бир
жа не работала в эти дни
dd=fill(dd,Ср.взв..ETC)
dd=fill(dd,Закр.)
dd=replace_na(dd,list('Объем.сделок.-.руб.'=0,'Сделок'=0))
dd=mutate(dd,Закр. = as.numeric(na_if(Закр., 0))) %>%  fill(Закр., .direction = 'up') # заменяем 0 на предыдущие значения
dd$regulations=ifelse(dd$Дата.торгов<as.Date('2014-11-10',origin = "1899-12-30"), 1, 0) # отказ от управляемого курса

oil <- read.xlsx("RBRTEd.xlsx",
                sheet = "Data 1")
oil$Sourcekey=as.Date(oil$Sourcekey,origin = "1899-12-30")

ir <- read.xlsx("ставка_цб.xlsx",
                sheet = "Sheet2")
ir$date=as.Date(ir$date,origin = "1899-12-30")

dd=merge(x = dd, y = oil, by.x = "Дата.торгов", by.y = 'Sourcekey', all.x = TRUE)
dd=merge(x = dd, y = ir, by.x = "Дата.торгов", by.y = 'date', all.x = TRUE)

dd=fill(dd,RBRTE)
dd=fill(dd,int_rate)

dd$int_rate=ifelse(is.na(dd$int_rate),18, dd$int_rate)



usd=ts(dd$Закр.,
       start = c(2003,105),
       frequency = 365)

autoplot(usd)
```



На графике временного ряда USDRUB_TOM видно, что это нестационарный процесс с возрастающим трендом и большими выбросами в конце ряда.

Проверка на стационарность

```
adf.test(usd)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  usd
## Dickey-Fuller = -3.3759, Lag order = 19, p-value = 0.05749
## alternative hypothesis: stationary
```

```
pp.test(usd)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  usd
## Dickey-Fuller Z(alpha) = -18.538, Truncation lag parameter = 11,
## p-value = 0.09378
## alternative hypothesis: stationary
```

```
kpss.test(usd)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  usd
## KPSS Level = 54.344, Truncation lag parameter = 11, p-value = 0.01
```

Стационарность на 5% уровне не подтвердилась в двух из трех тестов. Зато подтвердилась во всех на 10% уровне значимости.

```
adf.test(diff(usd)) #p-value = 0.01 - stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(usd)
## Dickey-Fuller = -18.825, Lag order = 19, p-value = 0.01
## alternative hypothesis: stationary
```

```
pp.test(diff(usd)) #p-value = 0.01 - stationary
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff(usd)
## Dickey-Fuller Z(alpha) = -7545, Truncation lag parameter = 11, p-value
## = 0.01
## alternative hypothesis: stationary
```

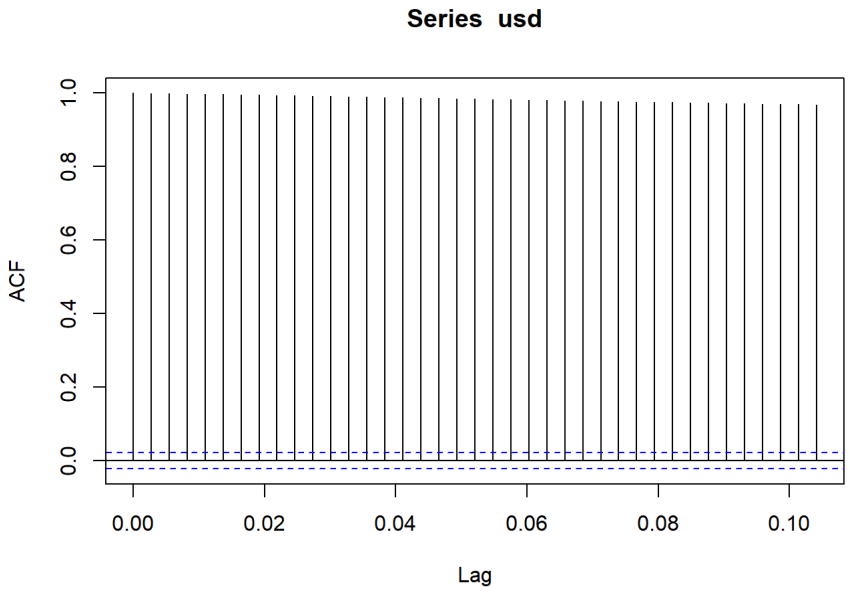
```
kpss.test(diff(usd)) #p-value = 0.0904 - stationary on 5% significance
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(usd)
## KPSS Level = 0.094636, Truncation lag parameter = 11, p-value = 0.1
```

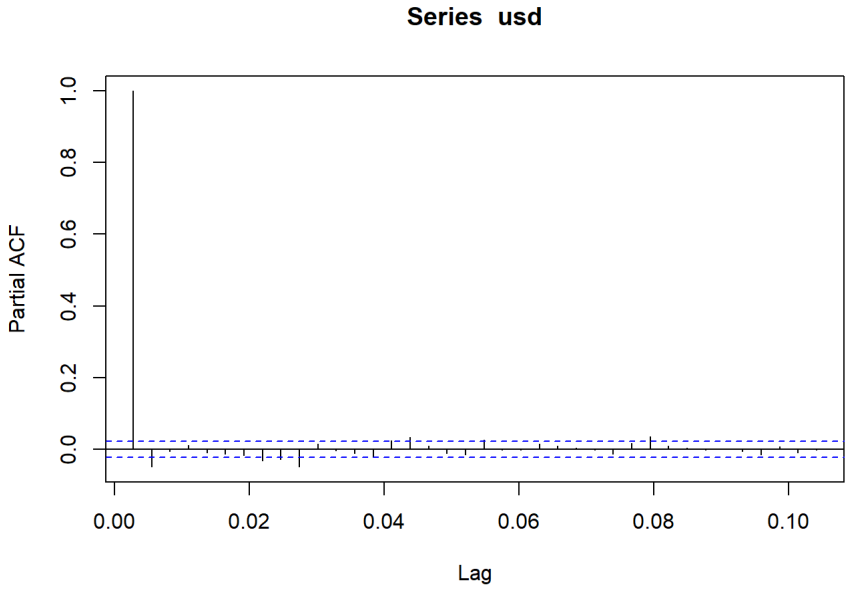
А в случае разницы значений, наоборот, подтвердилась в двух из трех тестов на 5% уровне и во всех на 10% уровне.

ACF PACF

```
acf(usd)
```



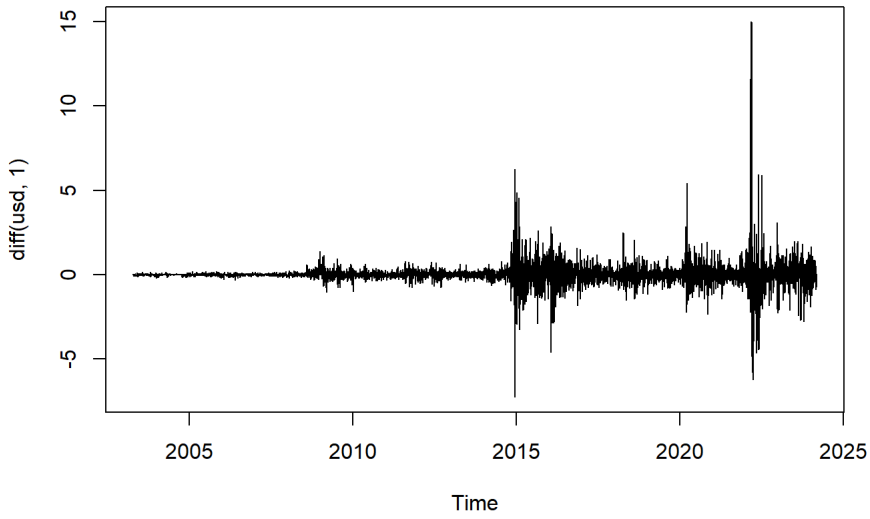

```
pacf(usd)
```



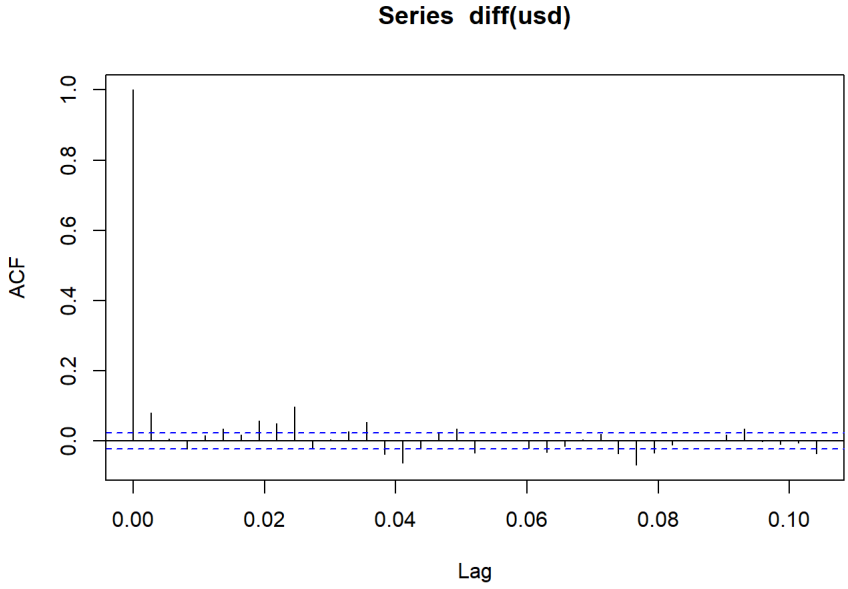
По графику ACF видно, что ряд нестационарен, т.к.автокорреляция с лагами везде превышает пороговое значение.

Чтобы преобразовать наш ряд в стационарный, возьмем первые разницы:

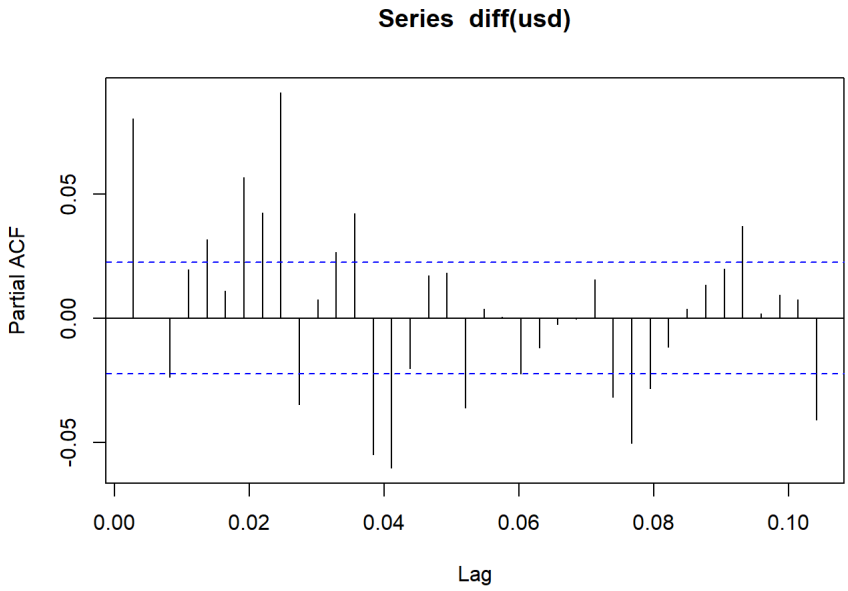
```
plot(diff(usd, 1))
```




```
acf(diff(usd))
```




```
pacf(diff(usd))
```




```
d_usd = diff(usd)
```

Мы использовали в работе

Модели

1. auto-ARIMA
2. другие аримы
3. ARIMAX
4. SARIMA
5. ETS
6. Исследовать на структурные сдвиги

В основе формирования моделей лежат следующие принципы: Мы решили кросс-валидировать ряд с шагом = 100. Обучить модель на тренировочной выборке данных (trainUSD для моделей, не требующих стационарности и d_trainUSD, требующих стационарности). Сформировать прогноз с шагом 1 на будущие 100 значений. Сравнить полученные результаты с тестовой выборкой (testUSD для моделей, не требующих стационарности и d_testUSD, требующих стационарности). Рассчитать основные показатели на основе ошибки прогноза: MSE, RMSE, MAE, MAPE, MASE.

Тип тестироавния

cross validation (fixed window) 7524/100

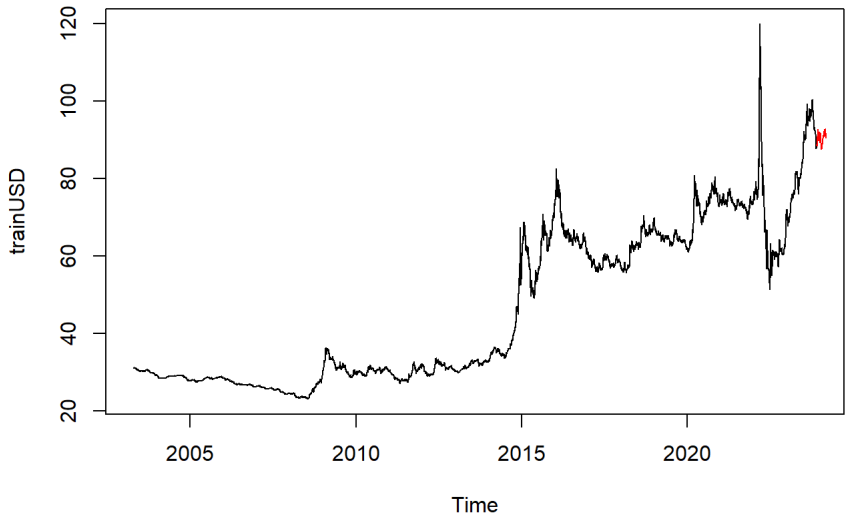
Метод тестирования

1. MASE
2. MSE
3. RMSE
4. MAE
5. MAPE
6. тест диболда-мариано

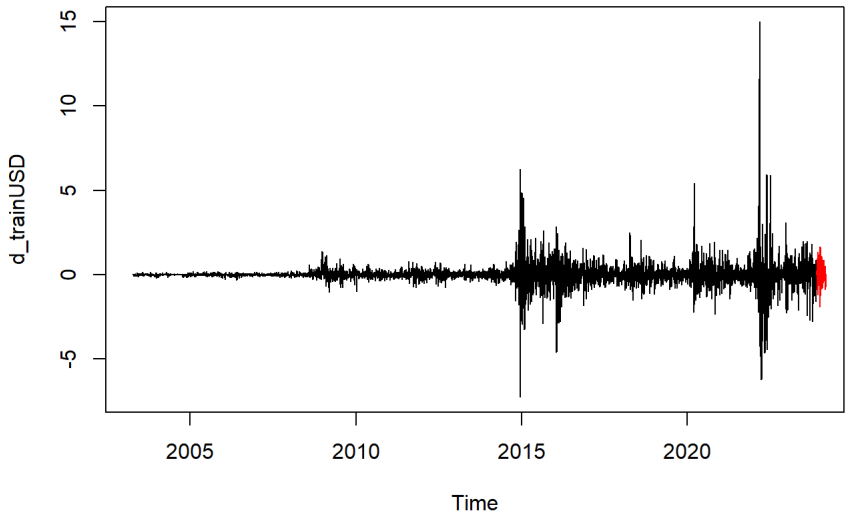
Выборки

Разделим ряд usd на тренировочную и тестовую выборку. К тестовой части будет относиться последние 100 значений ряда.

```
#nToPred = 0.1*(length(d_usd))
nToPred = 100
testUSD = ts(usd[(length(d_usd) - nToPred):length(usd)],
       end = c(2024,64), #end = c(2024,3, 5), start = c(2023,11,25),
       frequency = 365)# from 7524 to 7624

trainUSD = ts(usd[1:(length(usd) - nToPred - 1)],
       start = c(2003,104), #end = c(2023,11,25),
       frequency = 365) # from 1 to 7523

# Сформируем для обеих частей (тренировочной и тестовой) первые разности, чтобы далее исследовать более стационарный ряд

d_testUSD = diff(testUSD)
d_trainUSD = diff(trainUSD)

plot(trainUSD)
lines(testUSD, col = "red")
```




```
plot(d_trainUSD)
lines(d_testUSD, col = "red")
```



Наивная модель

```
#Naive-forecasting for I=0
err_Naive = c()

NaiveForecast = trainUSD[1:length(testUSD)]
err_Naive = testUSD - NaiveForecast

mean(err_Naive^2) ##MSE
```

```
## [1] 3587.766
```

```
sqrt(mean(err_Naive)) ##RMSE
```

```
## [1] 7.738247
```

```
mean(abs(err_Naive)) ##MAE
```

```
## [1] 59.88047
```

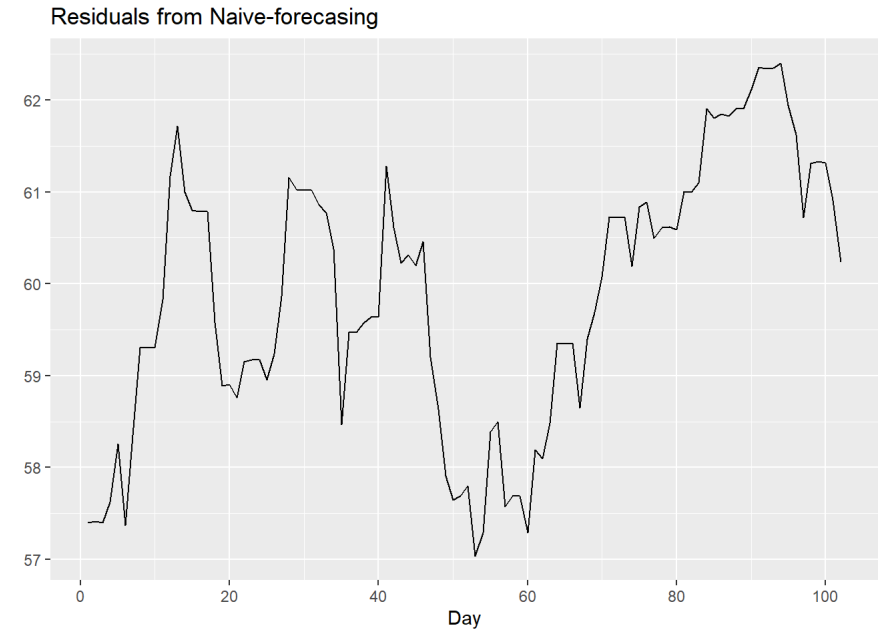
```
mean(abs(err_Naive/testUSD)*100) ##MAPE
```

```
## [1] 66.1051
```

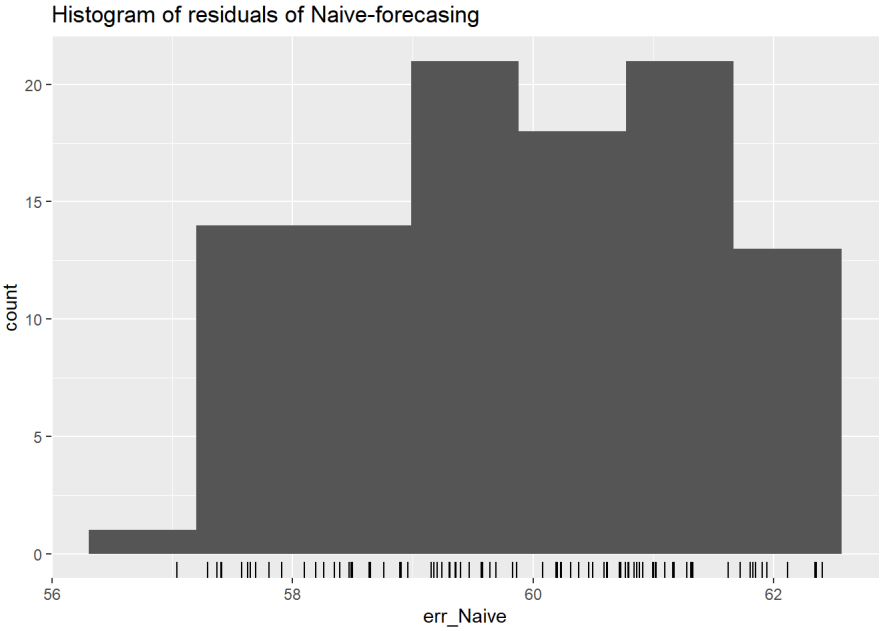
```
mean(abs(err_Naive))/mean(abs(err_Naive)) ##MASE
```

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

```
autoplot(ts(err_Naive)) + xlab("Day") + ylab("") +
  ggtitle("Residuals from Naive-forecasing")
```




```
gghistogram(err_Naive) + ggtitle("Histogram of residuals of Naive-forecasing")
```




```
#Naive-forecasting for I=1 (diff(usd))
err_Naive1 = c()

NaiveForecast = d_trainUSD[1:length(d_testUSD)]
err_Naive1 = d_testUSD - NaiveForecast

mean(err_Naive^2) ##MSE
```

```
## [1] 3587.766
```

```
sqrt(mean(err_Naive1)) ##RMSE
```

```
## [1] 0.1674503
```

```
mean(abs(err_Naive1)) ##MAE
```

```
## [1] 0.3751168
```

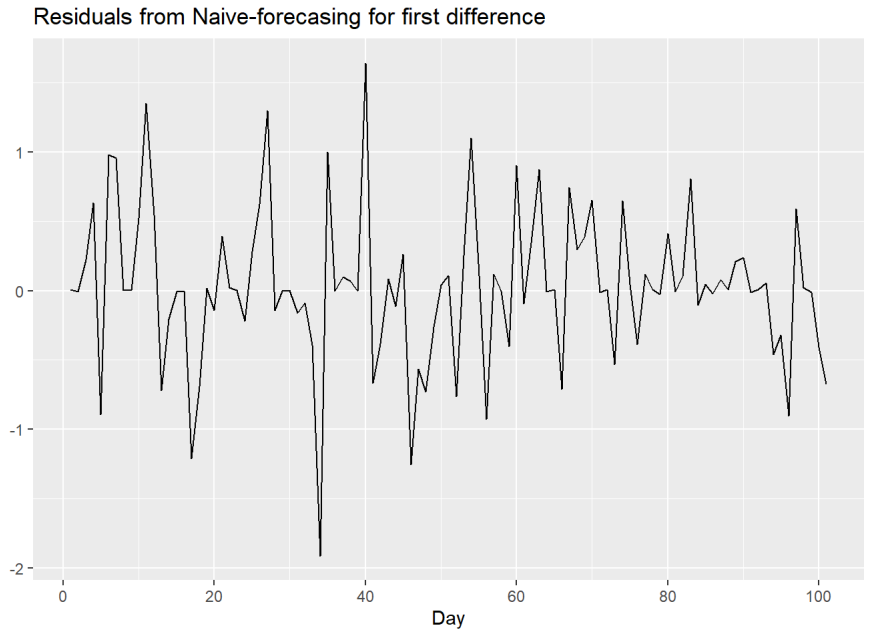
```
mean(abs(err_Naive1/testUSD)*100) ##MAPE
```

```
## [1] 0.4150678
```

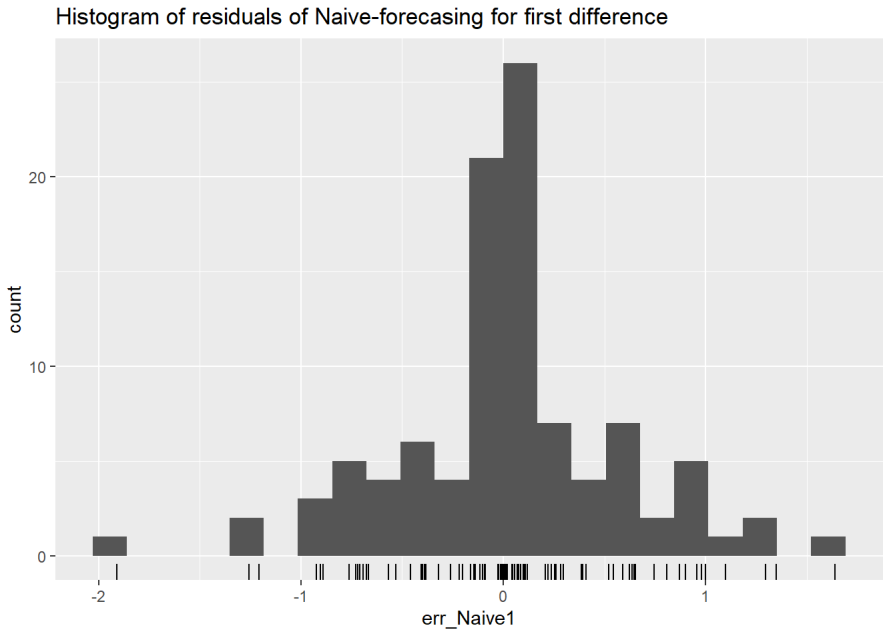
```
mean(abs(err_Naive1))/mean(abs(err_Naive1)) ##MASE
```

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

```
autoplot(ts(err_Naive1)) + xlab("Day") + ylab("") +
  ggtitle("Residuals from Naive-forecasing for first difference")
```




```
gghistogram(err_Naive1) + ggtitle("Histogram of residuals of Naive-forecasing for first difference")
```



Auto.ARIMA

Построим авто.ариму. За основу взят ряд с первыми разностями d_usd, поэтому в результате мы получили модель ARIMA(0,0,1) или модель скользящего среднего с одним лагом

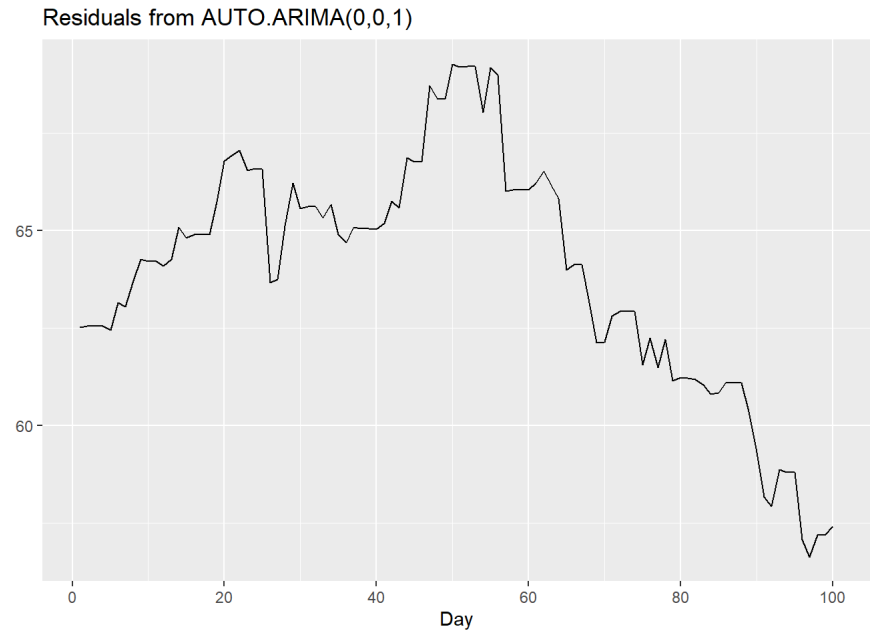
```
err_AUTO.ARIMA
```

```
##   [1] 62.51964 62.56105 62.55776 62.55802 62.45005 63.15761 63.05548 63.71665
##   [9] 64.26058 64.21731 64.22075 64.10443 64.27292 65.08271 64.80775 64.89978
##  [17] 64.89246 64.89304 65.73502 66.78567 66.93667 67.05960 66.54763 66.58853
##  [25] 66.58526 63.66982 63.75152 65.13007 66.22451 65.56393 65.61694 65.61268
##  [33] 65.32675 65.66837 64.89568 64.69836 65.08143 65.05072 65.05318 65.03678
##  [41] 65.19202 65.75208 65.59375 66.86471 66.76297 66.77110 68.71181 68.39189
##  [49] 68.38229 69.27390 69.20274 69.20842 69.20797 68.03914 69.18729 68.98842
##  [57] 66.01119 66.04552 66.04280 66.04302 66.21302 66.52605 66.16654 65.82807
##  [65] 63.98706 64.13381 64.12214 63.15679 62.12671 62.13620 62.81317 62.93719
##  [73] 62.92726 62.92806 61.55901 62.25429 61.48649 62.20580 61.14814 61.23160
##  [81] 61.22502 61.19317 61.05273 60.80218 60.84352 61.11538 61.09393 61.09562
##  [89] 60.36992 59.32644 58.17566 57.93526 58.87213 58.79750 58.80344 57.07556
##  [97] 56.63817 57.19411 57.20371 57.41891
```

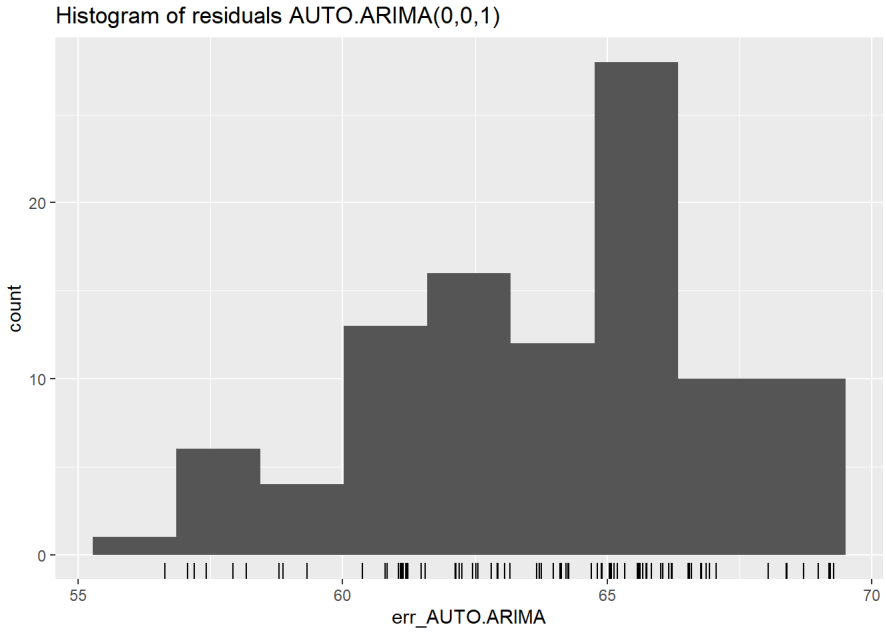
```
summary(mAUTO.ARIMA)
```

```
## Series: tmp
## ARIMA(0,1,1)
##
## Coefficients:
##          ma1
##       0.0798
## s.e.  0.0114
##
## sigma^2 = 0.2964:  log likelihood = -6019.82
## AIC=12043.64   AICc=12043.65   BIC=12057.47
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE      MAPE       MASE
## Training set 0.007270671 0.5443589 0.2186794 0.01033358 0.3962514 0.03520908
##                      ACF1
## Training set 0.0008051946
```

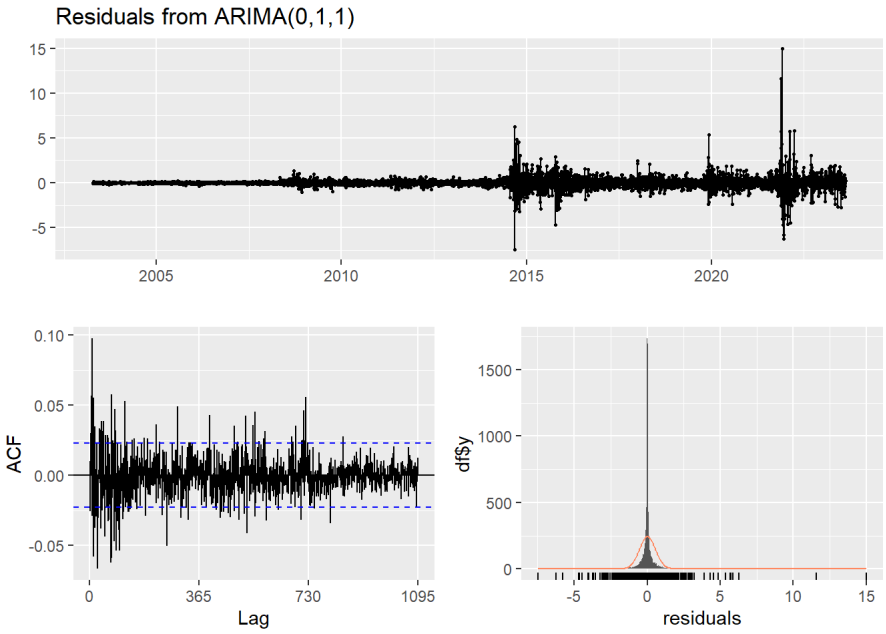
```
autoplot(ts(err_AUTO.ARIMA)) + xlab("Day") + ylab("") +
  ggtitle("Residuals from AUTO.ARIMA(0,0,1)")
```




```
gghistogram(err_AUTO.ARIMA) + ggtitle("Histogram of residuals AUTO.ARIMA(0,0,1)")
```




```
checkresiduals(mAUTO.ARIMA)
```



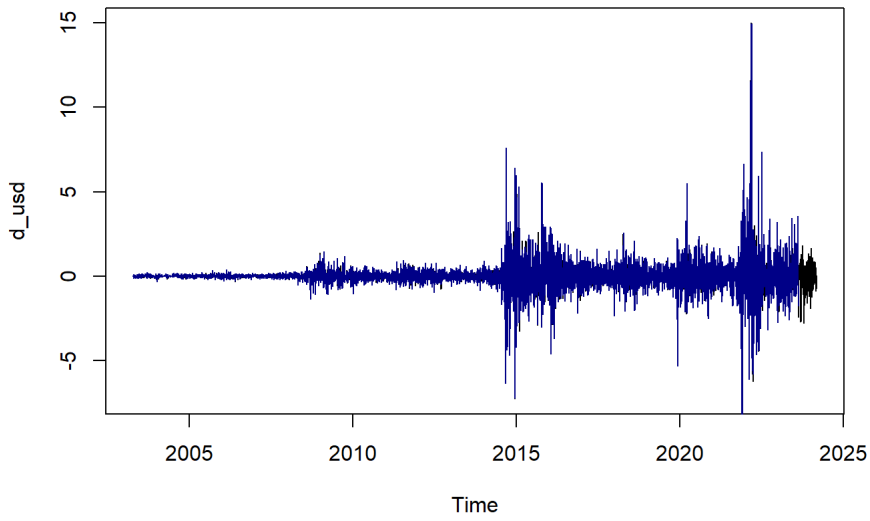

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)
## Q* = 1636.3, df = 729, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 730
```

Также нами было принято решение построить ARIMA(1,0,2), опираясь на графики ACF и PACF на основе ряда d_usd (с первыми разностями) MASE = 0.6239841

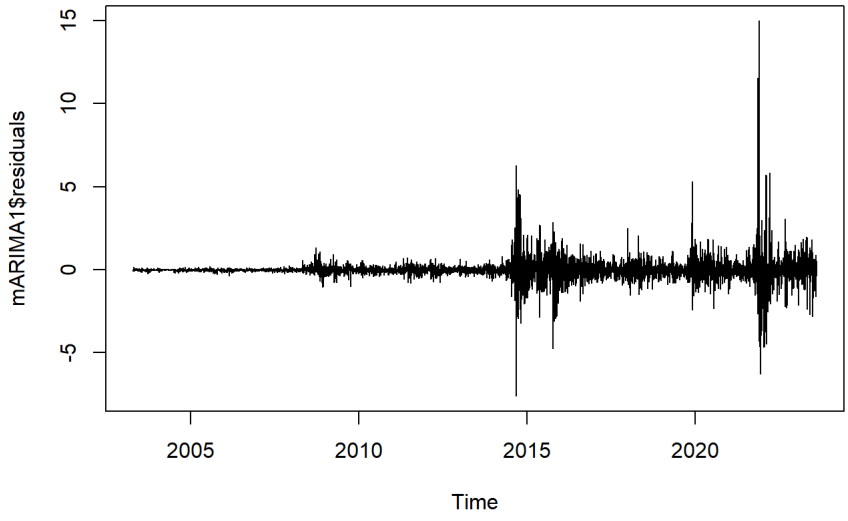
```
err_ARIMA1 = c()
for (i in 1:nCV){
  tmpUSD = d_trainUSD[(1+i-1):(length(d_trainUSD)-nCV+i)]  # fixed window
  tmp = ts(tmpUSD,
           start = c(2003,104),
           frequency = 365)

  mARIMA1 = Arima(tmp, order = c(1,0,2))
  err_ARIMA1[i] = forecast::forecast(mARIMA1)$mean[1] - d_trainUSD[1:(length(d_trainUSD)-i+1)]

}
summary(mARIMA1)
```

```
## Series: tmp
## ARIMA(1,0,2) with non-zero mean
##
## Coefficients:
##           ar1     ma1     ma2    mean
##       -0.4230  0.5049  0.0531  0.0078
## s.e.   0.1996  0.1994  0.0192  0.0069
##
## sigma^2 = 0.2963:  log likelihood = -6017.6
## AIC=12045.21   AICc=12045.22   BIC=12079.77
##
## Training set error measures:
##                        ME      RMSE       MAE MPE MAPE      MASE          ACF1
## Training set 4.475579e-05 0.5442328 0.2204888 NaN  Inf 0.6239841 -0.0008042921
```

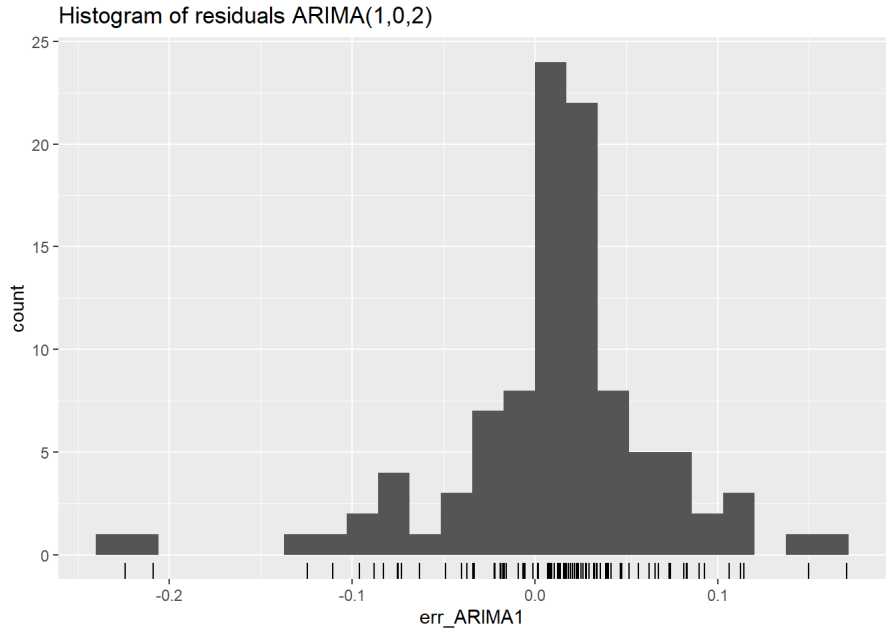
```
plot(d_usd)
lines((d_usd - mARIMA1$residuals), col='dark blue')
```



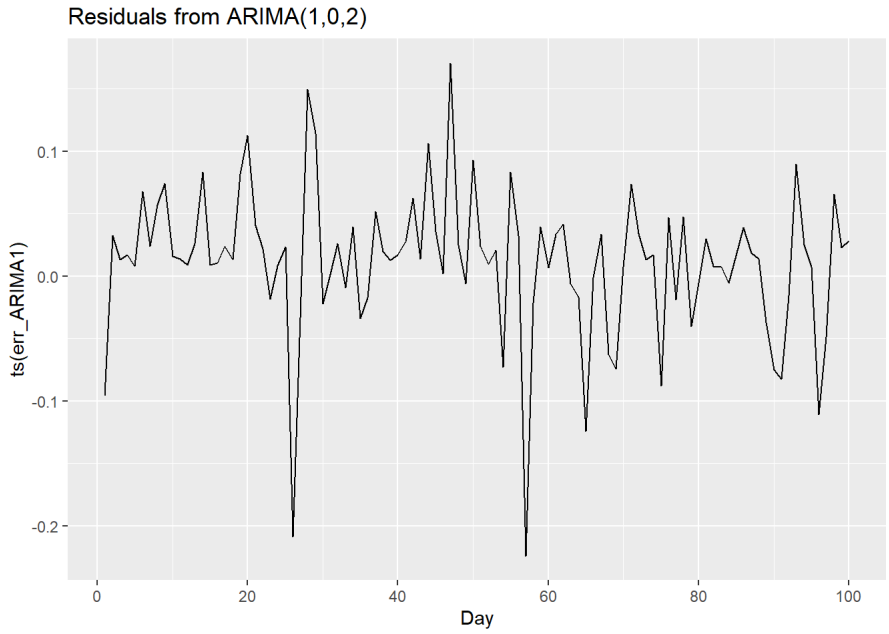

```
plot(mARIMA1$residuals)
```



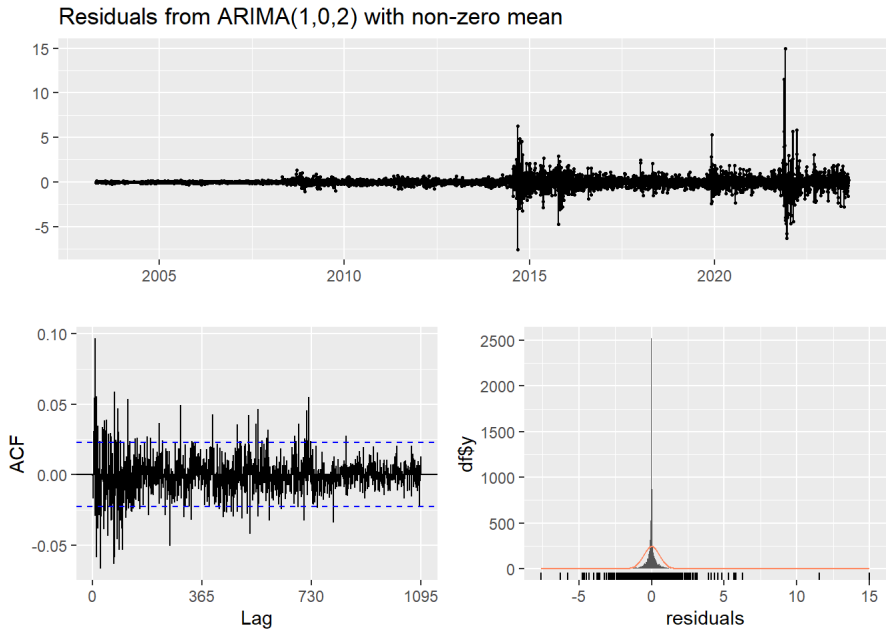

```
gghistogram(err_ARIMA1) + ggtitle("Histogram of residuals ARIMA(1,0,2)")
```




```
autoplot(ts(err_ARIMA1))+ xlab("Day") + ggtitle("Residuals from ARIMA(1,0,2)")
```




```
checkresiduals(mARIMA1)
```



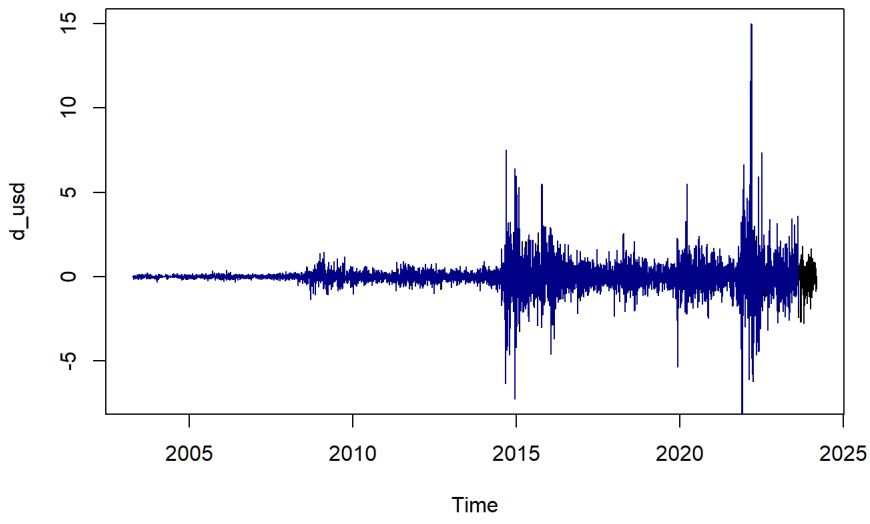
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,2) with non-zero mean
## Q* = 1633.3, df = 727, p-value < 2.2e-16
##
## Model df: 3.   Total lags used: 730
```

Вдобавок, нами было принято решение построить ARIMA(1,1,1), опираясь на графики ACF и PACF.В основе модели ряд d_usd (с первыми разностями), применяя модель ARIMA(1,1,1), мы хотели таким образом протестировать вторые разности.

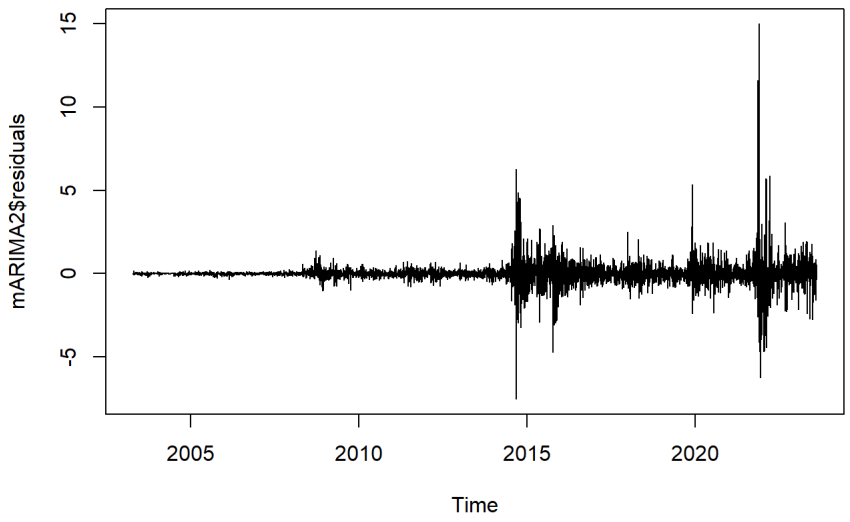
```
err_ARIMA2 = c()
for (i in 1:nCV){
  tmpUSD = d_trainUSD[(1+i-1):(length(d_trainUSD)-nCV+i)]  # fixed window
  tmp = ts(tmpUSD,
           start = c(2003,104),
           frequency = 365)

  mARIMA2 = Arima(tmp, order = c(1,1,1))
  err_ARIMA2[i] = forecast::forecast(mARIMA2)$mean[1] - d_trainUSD[1:(length(d_trainUSD)-i+1)]

}

summary(mARIMA2)
```

```
## Series: tmp
## ARIMA(1,1,1)
##
## Coefficients:
##          ar1     ma1
##       0.0811  -1e+00
## s.e.  0.0116   7e-04
##
## sigma^2 = 0.2964:  log likelihood = -6022.9
## AIC=12051.8   AICc=12051.8   BIC=12072.54
##
## Training set error measures:
##                       ME      RMSE       MAE MPE MAPE     MASE          ACF1
## Training set 0.005298404 0.5443236 0.2191743 NaN  Inf 0.620264 -0.0003394799
```

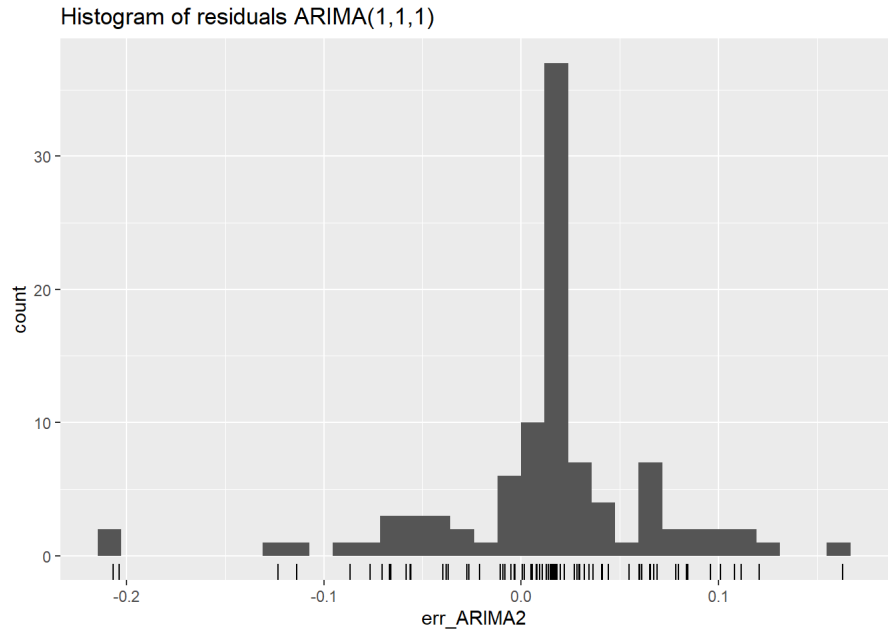
```
plot(d_usd)
lines((d_usd - mARIMA2$residuals), col='dark blue')
```



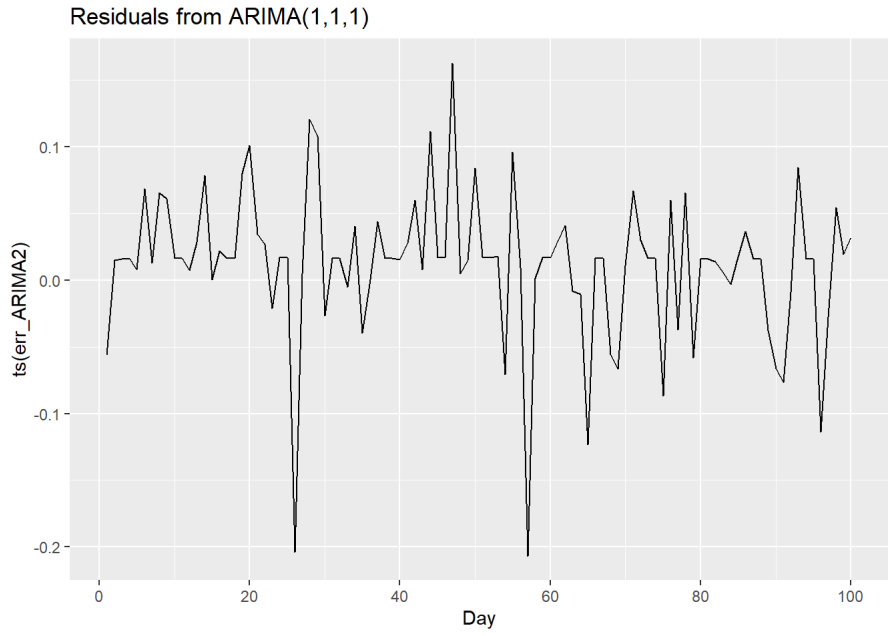

```
plot(mARIMA2$residuals)
```



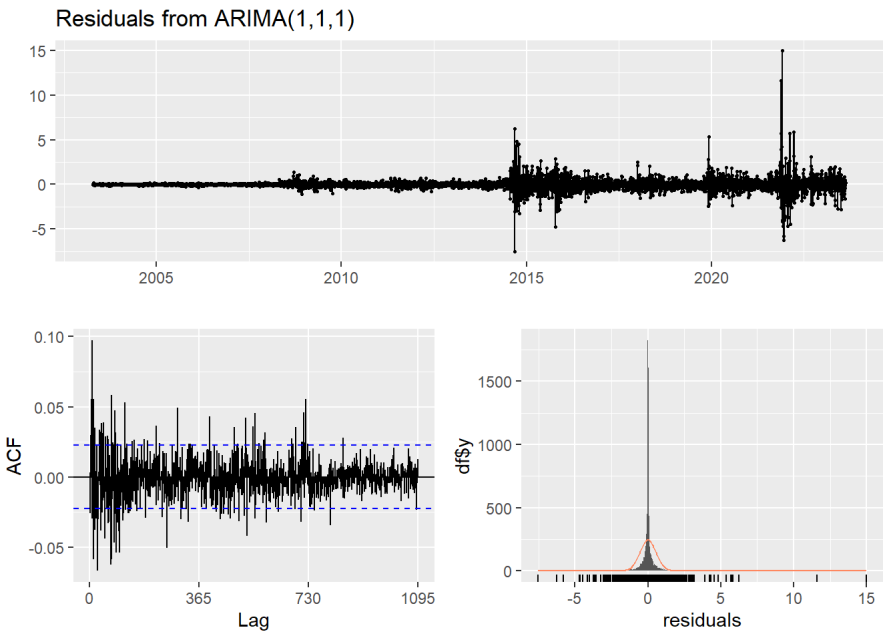

```
gghistogram(err_ARIMA2) + ggtitle("Histogram of residuals ARIMA(1,1,1)")
```




```
autoplot(ts(err_ARIMA2))+ xlab("Day") + ggtitle("Residuals from ARIMA(1,1,1)")
```



```
checkresiduals(mARIMA2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)
## Q* = 1637.1, df = 728, p-value < 2.2e-16
##
## Model df: 2.   Total lags used: 730
```

Результаты MASE:

ARIMA1(1,0,2) MASE = 0.6239841 (summary)

ARIMA2(1,1,1) MASE = 0.620264 (summary)

ARIMAX

Происходил выбор из следующих параметров: p - 0,1,3,7; d - 0,1,2; q - 0,1,3,7;

В качестве X были выбрана цена барреля нефти Брент, ключевая ставка, объемы торгов, дамми по работе биржи.

На тренировочной выборке кроссвалидацией с окном 100 была выбрана модель со следующими характеристиками: По MASE:

X1	X2	X3	X4	X5	X6	X7	X8
1	1	3	20	0.0499938	0.0024994	0.0499938	0.5705639
0	1	3	20	0.3823538	0.0386653	0.1966349	0.6682554
1	1	1	20	0.3803369	0.0381491	0.1953178	0.6685305
3	0	0	20	0.3809566	0.0384041	0.1959695	0.6686682
0	0	3	20	0.3811992	0.0384020	0.1959643	0.6686697

По RMSE:

X1	X2	X3	X4	X5	X6	X7	X8
3	1	7	100	0.0199724	0.0018611	0.0431400	0.7012913
1	1	3	20	0.0499938	0.0024994	0.0499938	0.5705639
3	0	7	100	0.0172654	0.0052619	0.0725392	0.6778840
7	0	7	100	0.0099805	0.0056706	0.0753032	0.7220237
0	2	3	100	0.0699770	0.0109201	0.1044994	0.6947131

При прогнозировании тестовой выборки с выбранным горизонтом прогноза получаем следующие ошибки: MASE

	X8
result.1	NA
result.2	NA
result.3	NA
result.4	NA

	X8
result.5	NA
result.6	as.data.frame.default(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
result.7	as.data.frame.default(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
result.8	as.data.frame.default(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)
result.9	as.data.frame.default(x[[i]], optional = TRUE, stringsAsFactors = stringsAsFactors)

При прогнозировании на 100 точек тестовой выборки получаем следующие ошибки: MASE

	message	call
result.1	система вычислительно сингулярная: величина, обратная к числу обусловленности, равна 1.44509e-16	hessian * n.used, A) | |result.2 |объект 'err_ARIMAX_100' не найден |eval(quote({ ...future.makeSendCondition <- base::local({ sendCondition <- NULL function(frame = 1) { if (is.function(sendCondition)) return(sendCondition) ns <- getNamespace("parallel") if (exists("sendData", mode = "function", envir = ns)) { parallel_sendData <- get("sendData", mode = "function", envir = ns) envir <- sys.frame(frame) master <- NULL while (!identical(envir, .GlobalEnv) && !identical(envir, emptyenv())) { if (exists("master", mode = "list", envir = envir, inherits = FALSE)) { master <- get("master", mode = "list", envir = envir, inherits = FALSE) if (inherits(master, c("SOCKnode", "SOCK0node"))) { sendCondition <<- function(cond) { data <- list(type = "VALUE", value = cond, success = TRUE) parallel_sendData(master, data) } return(sendCondition) } } frame <- frame + 1 envir <- sys.frame(frame) } } sendCondition <<- function(cond) NULL } }) withCallingHandlers({ { lapply(seq_along(...future.x_ii), FUN = function(jj) { ...future.x_jj <- ...future.x_ii[[jj]] { NULL k <- NULL } ...future.env <- environment() local({ for (name in names(...future.x_jj)) { assign(name, ...future.x_jj[[name]], envir = ...future.env, inherits = FALSE) } }) tryCatch({ order = c(all_res_best[all_res_best

solve.default(res X4 == 100,][k, “X1”], all_res_best[all_res_best$X4 == 100,][k, ” X2 ”], $all_res_best[all_res_best X4 == 100,][k, “X3”])

```
            m_arimax = Arima(d_trainUSD, order = order, xreg = data.matrix(trainX))
            err_ARIMAX_100[nrow(err_ARIMAX_100) + 1, ] = mean(forecast::forecast(m_arimax, 100, xreg = data.matrix(testX))
$mean - d_testUSD)
            mase_ARIMAX_100[nrow(mase_ARIMAX_100) + 1, ] = Metrics::mase(d_testUSD, forecast::forecast(m_arimax, 100, xreg
= data.matrix(testX))$mean)
            return(mase_ARIMAX_100)
          }, error = identity)
        })
    }
}, immediateCondition = function(cond) {
    sendCondition <- ...future.makeSendCondition()
    sendCondition(cond)
    muffleCondition <- function (cond, pattern = "^muffle")
    {
        inherits <- base::inherits
        invokeRestart <- base::invokeRestart
        is.null <- base::is.null
        muffled <- FALSE
        if (inherits(cond, "message")) {
            muffled <- grepl(pattern, "muffleMessage")
            if (muffled) invokeRestart("muffleMessage")
        } else if (inherits(cond, "warning")) {
            muffled <- grepl(pattern, "muffleWarning")
            if (muffled) invokeRestart("muffleWarning")
        } else if (inherits(cond, "condition")) {
            if (!is.null(pattern)) {
              computeRestarts <- base::computeRestarts
              grepl <- base::grepl
              restarts <- computeRestarts(cond)
              for (restart in restarts) {
                name <- restart$name
                if (is.null(name)) next
                if (!grepl(pattern, name)) next
                invokeRestart(restart)
                muffled <- TRUE
                break
              }
            }
        }
        invisible(muffled)
    }
    muffleCondition(cond)
})
```

}), new.env()) | |result.3 |система вычислительно сингулярная: величина, обратная к числу обусловленности, равна 1.76298e-16 hessian * n.used, A) | |result.4 |объект 'err_ARIMAX_100' не найден |eval(quote({ ...future.makeSendCondition <- base::local({ sendCondition <- NULL function(frame = 1) { if (is.function(sendCondition)) return(sendCondition) ns <- getNamespace("parallel") if (exists("sendData", mode = "function", envir = ns)) { parallel_sendData <- get("sendData", mode = "function", envir = ns) envir <- sys.frame(frame) master <- NULL while (!identical(envir, .GlobalEnv) && !identical(envir, emptyenv())) { if (exists("master", mode = "list", envir = envir, inherits = FALSE)) { master <- get("master", mode = "list", envir = envir, inherits = FALSE) if (inherits(master, c("SOCKnode", "SOCK0node"))) { sendCondition <<- function(cond) { data <- list(type = "VALUE", value = cond, success = TRUE) parallel_sendData(master, data) } return(sendCondition) } } frame <- frame + 1 envir <- sys.frame(frame) } } sendCondition <<- function(cond) NULL } }) withCallingHandlers({ { lapply(seq_along(...future.x_ii), FUN = function(jj) { ...future.x_jj <- ...future.x_ii[[jj]] { NULL k <- NULL } ...future.env <- environment() local({ for (name in names(...future.x_jj)) { assign(name, ...future.x_jj[[name]], envir = ...future.env, inherits = FALSE) } }) tryCatch({ order = c(all_res_best[all_res_best |solve.default(res X4 == 100,][k, “X1”], all_res_best[all_res_best $X4 == 100,][k, ” X2 ”], all_r e s b e s t X4 == 100,][k, “X3”]) m_arimax = Arima(d_trainUSD, order = order, xreg = data.matrix(trainX)) err_ARIMAX_100[nrow(err_ARIMAX_100) + 1,] = mean(forecast::forecast(m_arimax, 100, xreg = data.matrix(testX)) $mean - d t e s t U S D) m a s e A R I M A X 1 00[nrow(m a s e A R I M A X 1 00) + 1,] = M e t r i c s :: m a s e (d t e s t U S D , f o r e c a s t :: f o r e c a s t (m a r i m a x , 10 mean) return(mase_ARIMAX_100) }, error = identity) }) } }, immediateCondition = function(cond) { sendCondition <- …future.makeSendCondition() sendCondition(cond) muffleCondition <- function (cond, pattern = “^muffle”) { inherits <- base::inherits invokeRestart <- base::invokeRestart is.null <- base::is.null muffled <- FALSE if (inherits(cond, “message”)) { muffled <- grepl(pattern, “muffleMessage”) if (muffled) invokeRestart(“muffleMessage”) } else if (inherits(cond, “warning”)) { muffled <- grepl(pattern, “muffleWarning”) if (muffled) invokeRestart(“muffleWarning”) } else if (inherits(cond, “condition”)) { if (!is.null(pattern)) { computeRestarts <- base::computeRestarts grepl <- base::grepl restarts <- computeRestarts(cond) for (restart in restarts) { name <- restart name if (is.null(name)) next if (!grepl(pattern, name)) next invokeRestart(restart) muffled <- TRUE break } } } invisible(muffled) } muffleCondition(cond) }) }), new.env()) | |result.5 |пропущенное значение, а нужно TRUE/FALSE |if (length(x) <= order[2] + seasonal order[2] * seasonal $p e r i o d) s t o p (” N o t e n o u g h d a t a t o f i t t h e m o d e l ”) | | r e s u l t . 6 | пропущенное значение, а нужно T R U E / F A L S E | i f (l e n g t h (x) <= o r d e r [2] + s e a s o n a l order[2] * seasonal $p e r i o d) s t o p (” N o t e n o u g h d a t a t o f i t t h e m o d e l ”) | | r e s u l t . 7 | пропущенное значение, а нужно T R U E / F A L S E | i f (l e n g t h (x) <= o r d e r [2] + s e a s o n a l order[2] * seasonal $p e r i o d) s t o p (” N o t e n o u g h d a t a t o f i t t h e m o d e l ”) | | r e s u l t . 8 | пропущенное значение, а нужно T R U E / F A L S E | i f (l e n g t h (x) <= o r d e r [2] + s e a s o n a l order[2] * seasonal$period) stop(“Not enough data to fit the model”) |

SARIMAX

В рамках моделирования с помощью SARIMAX были проверены параметры сезонности от 0 до 2 для лучших моделей ARIMAX. Однако выявить релевантные параметры для данных моделей не удалось

ETS - (Errors, Trend, Seasonal) (без учета первых разниц)

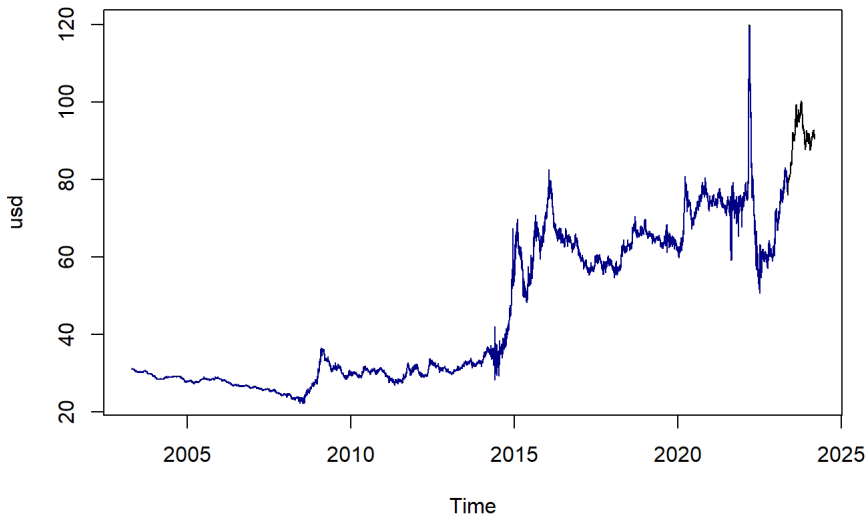
```
## MASE = 0.03547228
err_ETS= c()

for (i in 1:nCV){
  tmpUSD = trainUSD[(1+i-1):(length(trainUSD)-nCV+i)]  # fixed window
  tmp = ts(tmpUSD,
           start = c(2003,04,15),
           frequency = 365)

  mETS = ets(tmp)
  err_ETS[i] = forecast::forecast(mETS)$mean[1] - usd[(length(usd)-nToPred+1)]

}

err_ETS
```

```
##   [1] -88.59092 -88.59110 -88.59104 -88.59096 -88.59097 -88.59177 -88.59087
##   [8] -88.59095 -88.59061 -88.59060 -88.59065 -88.59071 -88.59061 -88.59056
##  [15] -88.59064 -88.59071 -88.59071 -88.59069 -88.59050 -88.59024 -88.59024
##  [22] -88.58985 -88.59030 -88.59035 -88.59035 -88.59075 -88.59080 -88.59061
##  [29] -88.58980 -88.58985 -88.59038 -88.59051 -88.59054 -88.59052 -88.59051
##  [36] -88.59060 -88.59059 -88.59058 -88.59051 -88.59012 -88.59056 -88.59121
##  [43] -88.59041 -88.59016 -88.59030 -88.59031 -88.58999 -88.59007 -88.59007
##  [50] -88.58979 -88.59008 -88.58985 -88.58991 -88.59009 -88.58988 -88.58993
##  [57] -88.59027 -88.59036 -88.59036 -88.59031 -88.59041 -88.59023 -88.59044
##  [64] -88.59039 -88.59068 -88.59066 -88.59076 -88.59075 -88.59098 -88.59097
##  [71] -88.59089 -88.59091 -88.59079 -88.59086 -88.59091 -88.59099 -88.59107
##  [78] -88.59097 -88.59051 -88.59185 -88.59101 -88.59102 -88.59095 -88.59106
##  [85] -88.59120 -88.59112 -88.59102 -88.59113 -88.59119 -88.59142 -88.59160
##  [92] -88.59165 -88.59140 -88.59155 -88.59148 -88.59177 -88.59202 -88.59180
##  [99] -88.59180 -88.59180
```

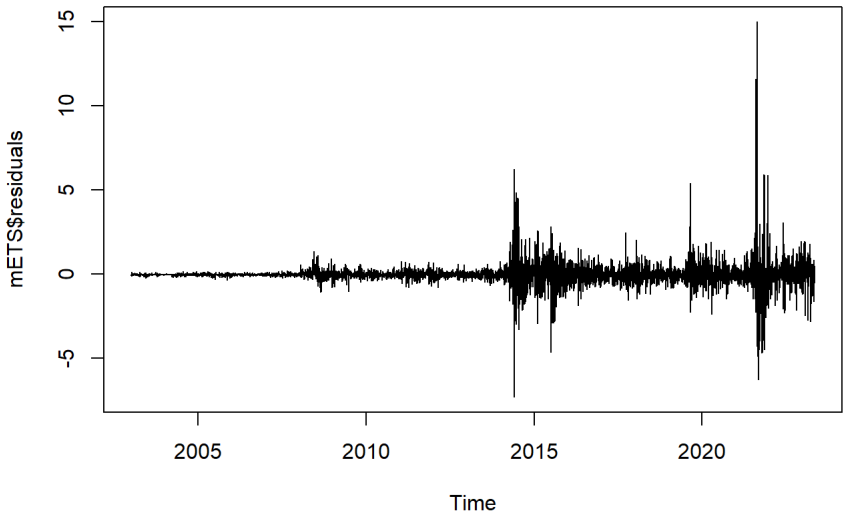
```
summary(mETS)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = tmp)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 0.0072
##
##   sigma:  0.5462
##
##      AIC     AICc      BIC
## 57190.93 57190.93 57211.67
##
## Training set error measures:
##                       ME      RMSE       MAE  MPE MAPE     MASE       ACF1
## Training set 0.001351863 0.5461394 0.2164722 -Inf  Inf 0.612617 0.08090512
```

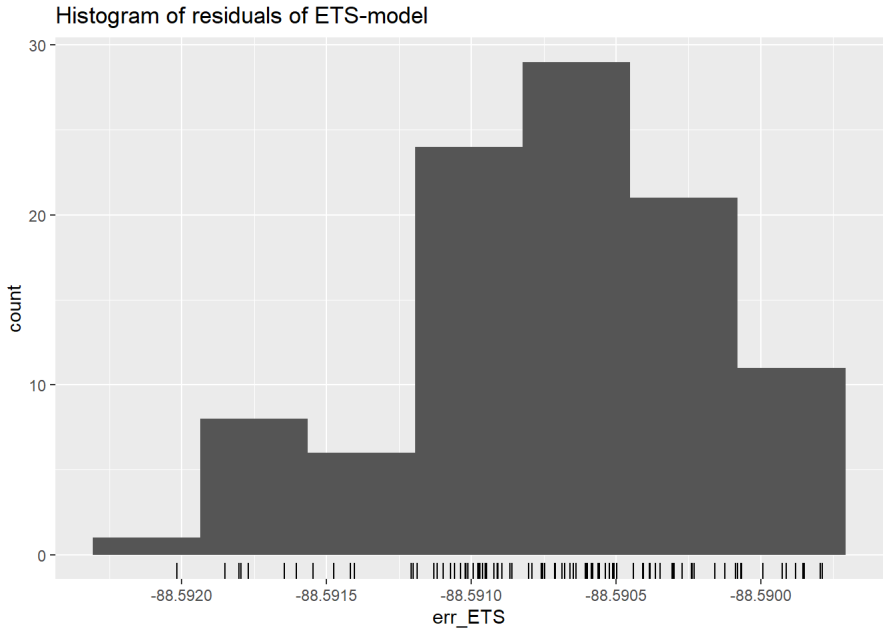
```
plot(usd)
lines((usd - mETS$residuals), col='dark blue')
```



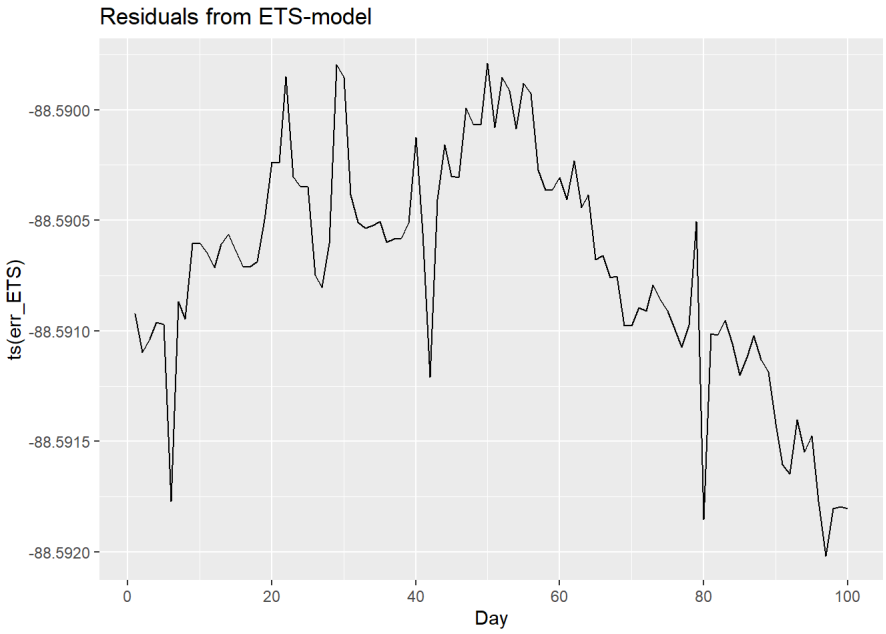

```
plot(mETS$residuals)
```



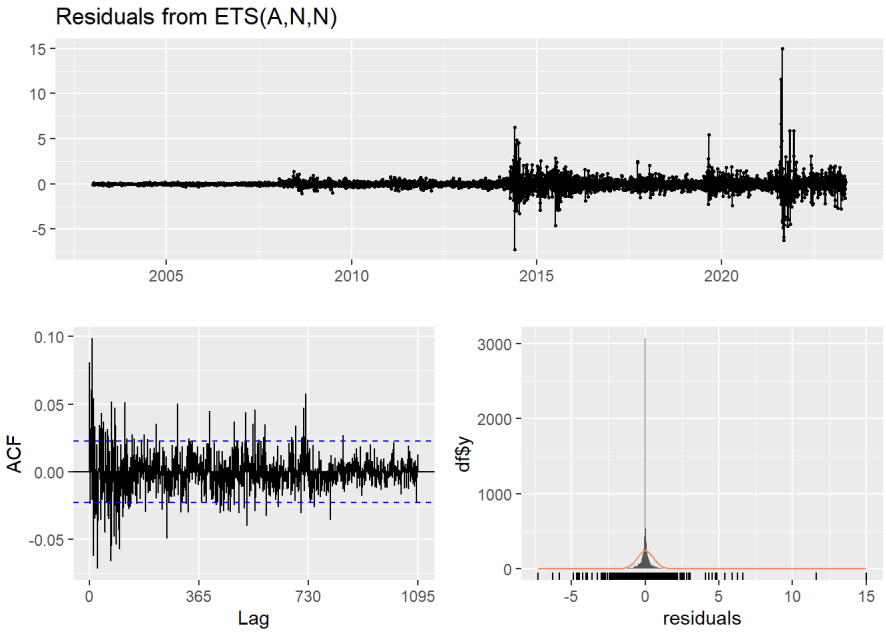

```
gghistogram(err_ETS) + ggtitle("Histogram of residuals of ETS-model")
```




```
autoplot(ts(err_ETS))+ xlab("Day") + ggtitle("Residuals from ETS-model")
```




```
checkresiduals(mETS)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 1742.7, df = 730, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 730
```

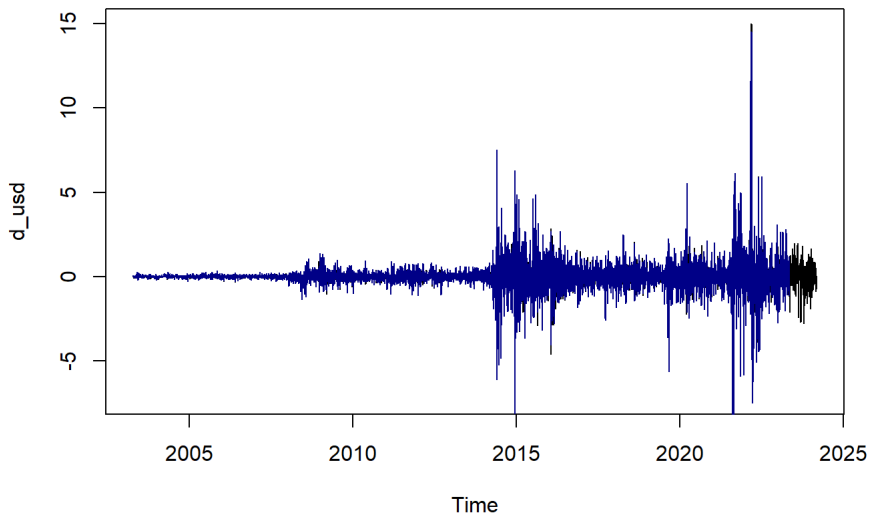
```
##ETS - model (Errors, Trend, Seasonal) (with first differences) (с первыми разницами)
## MASE = 1.479459
err_ETS1= c()

for (i in 1:nCV){
  tmpUSD = d_trainUSD[(1+i-1):(length(d_trainUSD)-nCV+i)]  # fixed window
  tmp = ts(tmpUSD,
           start = c(2003,04,15),
           frequency = 365)

  mETS1 = ets(tmp)
  err_ETS1[i] = forecast::forecast(mETS1)$mean[1] - usd[(length(d_usd)-nToPred+1)]

}

err_ETS1
```

```
##   [1] -88.59092 -88.59110 -88.59104 -88.59096 -88.59097 -88.59177 -88.59087
##   [8] -88.59095 -88.59061 -88.59060 -88.59065 -88.59071 -88.59061 -88.59056
##  [15] -88.59064 -88.59071 -88.59071 -88.59069 -88.59050 -88.59024 -88.59024
##  [22] -88.58985 -88.59030 -88.59035 -88.59035 -88.59075 -88.59080 -88.59061
##  [29] -88.58980 -88.58985 -88.59038 -88.59051 -88.59054 -88.59052 -88.59051
##  [36] -88.59060 -88.59059 -88.59058 -88.59051 -88.59012 -88.59056 -88.59121
##  [43] -88.59041 -88.59016 -88.59030 -88.59031 -88.58999 -88.59007 -88.59007
##  [50] -88.58979 -88.59008 -88.58985 -88.58991 -88.59009 -88.58988 -88.58993
##  [57] -88.59027 -88.59036 -88.59036 -88.59031 -88.59041 -88.59023 -88.59044
##  [64] -88.59039 -88.59068 -88.59066 -88.59076 -88.59075 -88.59098 -88.59097
##  [71] -88.59089 -88.59091 -88.59079 -88.59086 -88.59091 -88.59099 -88.59107
##  [78] -88.59097 -88.59051 -88.59185 -88.59101 -88.59102 -88.59095 -88.59106
##  [85] -88.59120 -88.59112 -88.59102 -88.59113 -88.59119 -88.59142 -88.59160
##  [92] -88.59165 -88.59140 -88.59155 -88.59148 -88.59177 -88.59202 -88.59180
##  [99] -88.59180 -88.59180
```

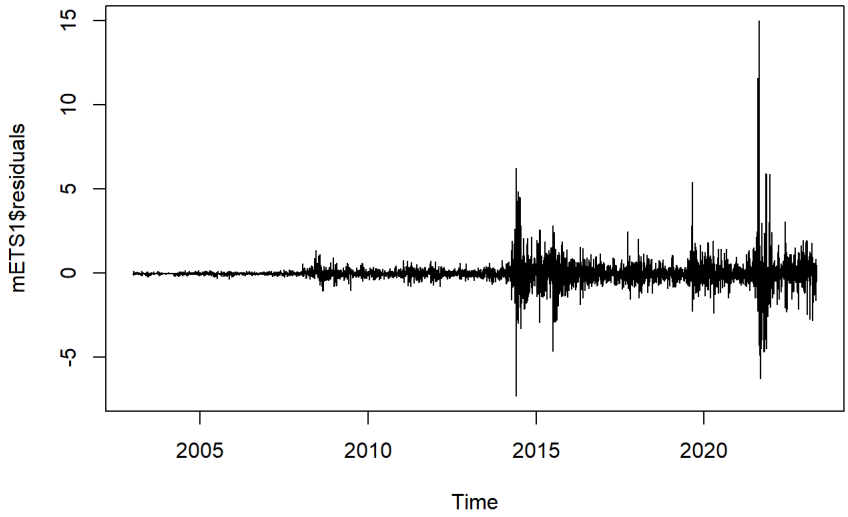
```
summary(mETS1)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = tmp)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 0.0072
##
##   sigma:  0.5462
##
##      AIC     AICc      BIC
## 57190.93 57190.93 57211.67
##
## Training set error measures:
##                       ME      RMSE       MAE  MPE MAPE     MASE       ACF1
## Training set 0.001351863 0.5461394 0.2164722 -Inf  Inf 0.612617 0.08090512
```

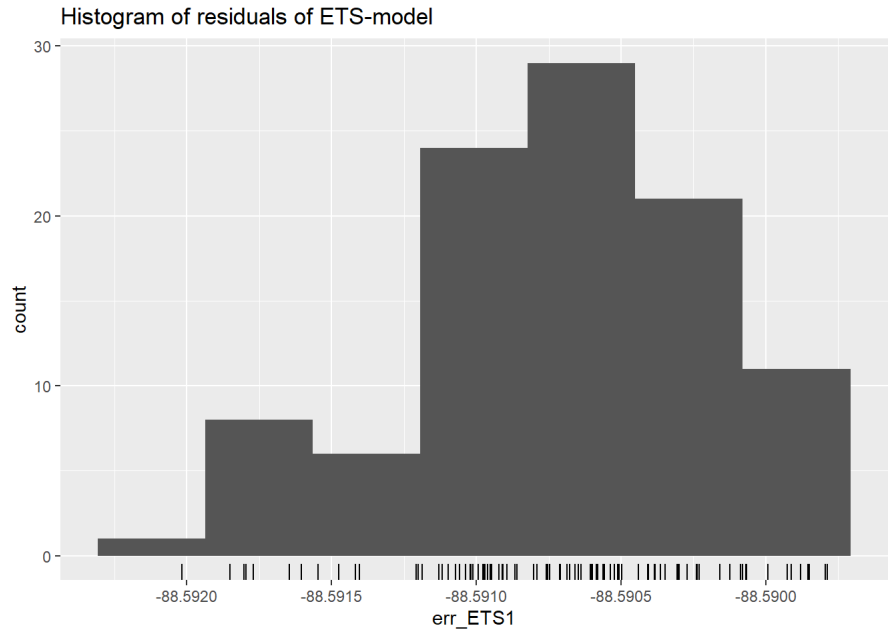
```
plot(d_usd)
lines((d_usd - mETS1$residuals), col='dark blue')
```



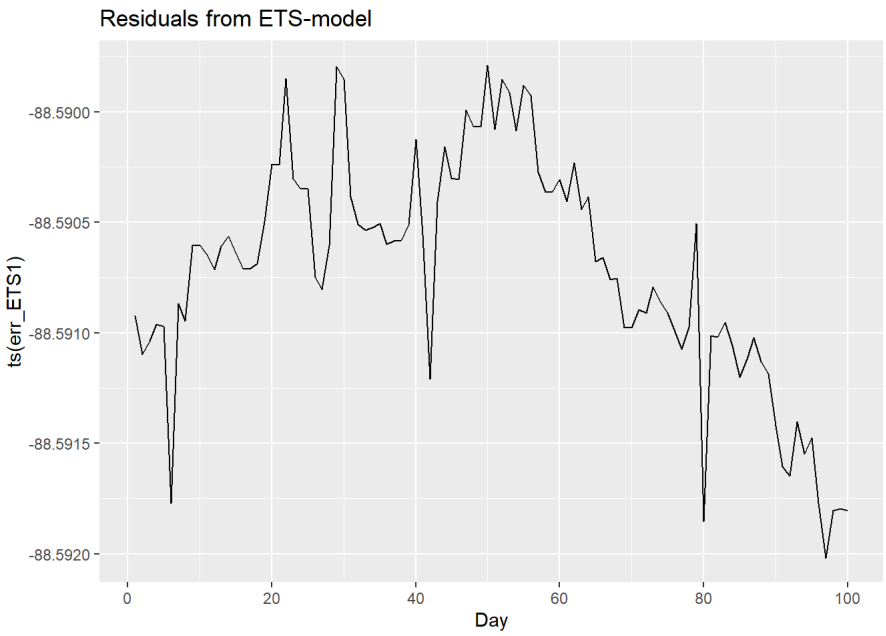

```
plot(mETS1$residuals)
```



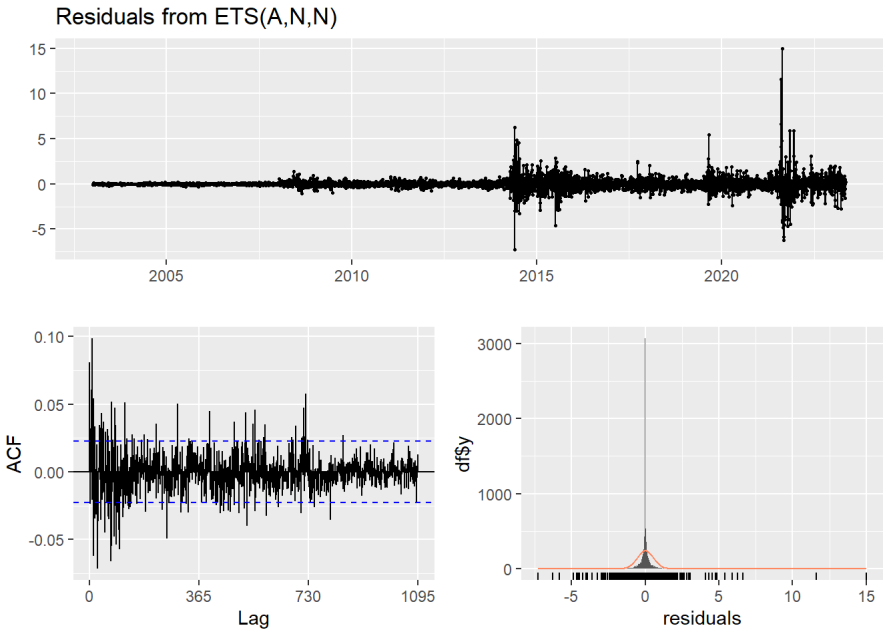

```
gghistogram(err_ETS1) + ggtitle("Histogram of residuals of ETS-model")
```



```
autoplot(ts(err_ETS1))+ xlab("Day") + ggtitle("Residuals from ETS-model")
```




```
checkresiduals(mETS1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,N)
## Q* = 1742.7, df = 730, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 730
```

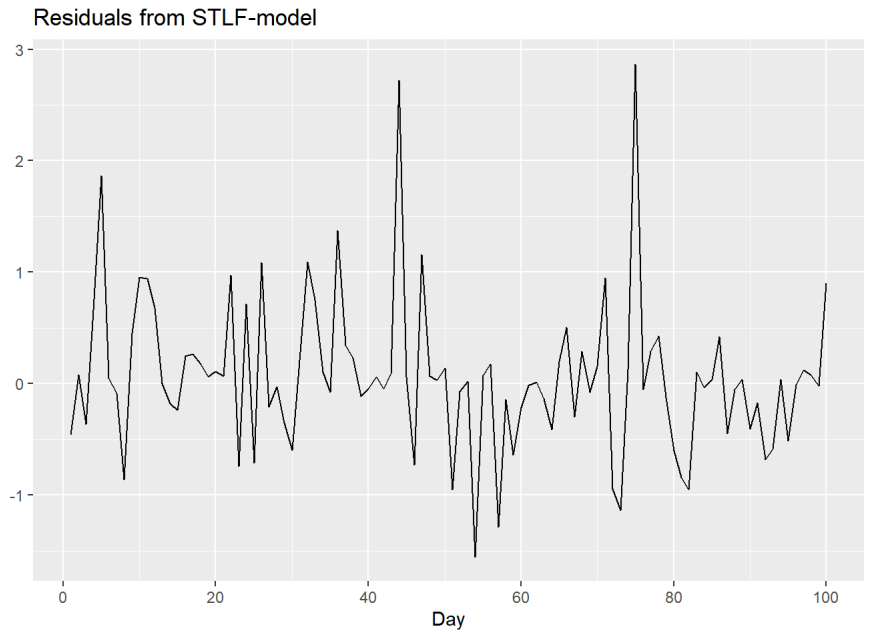
```
##STLF (Seasonal and Trend decomposition using Loess) - метод основан на локальной регрессии (Loess)
## MASE = 0.03711753
err_STLF= c()

for (i in 1:nCV){
  tmpUSD = trainUSD[(1+i-1):(length(trainUSD)-nCV+i)]  # fixed window
  tmp = ts(tmpUSD,
           start = c(2003,04,15),
           frequency = 365)

  mSTLF <- stlf(tmp)
  err_STLF[i] = forecast::forecast(mSTLF)$mean[1] - trainUSD[(length(trainUSD)-i+1)]
}

err_STLF
```

```
##   [1] -0.460726018  0.076686493 -0.367048241  0.658970104  1.866880131
##   [6]  0.046367466 -0.086473820 -0.863300102  0.453636481  0.952467598
##  [11]  0.943041321  0.680132574  0.004097342 -0.178700303 -0.239088495
##  [16]  0.244738921  0.267764719  0.173101349  0.060305017  0.106819340
##  [21]  0.066719022  0.973179554 -0.742552005  0.716713333 -0.709464047
##  [26]  1.086423413 -0.207455434 -0.029929322 -0.351187697 -0.595259567
##  [31]  0.212155047  1.089149336  0.752650155  0.100880197 -0.077787767
##  [36]  1.373430343  0.345920255  0.227529490 -0.115078794 -0.047722759
##  [41]  0.064154199 -0.044816164  0.091006758  2.723049834  0.063434041
##  [46] -0.729099669  1.155023706  0.064669648  0.032562450  0.139514773
##  [51] -0.951729529 -0.068190292  0.018453977 -1.553854852  0.071169042
##  [56]  0.177189486 -1.289783978 -0.142671045 -0.642454881 -0.217129594
##  [61] -0.019147159  0.013485211 -0.139340811 -0.413249308  0.200302277
##  [66]  0.507236575 -0.295364681  0.286169752 -0.076565218  0.157982928
##  [71]  0.950100743 -0.947898417 -1.136639713  0.129518656  2.868095390
##  [76] -0.054920372  0.290092651  0.428133473 -0.125162006 -0.601889981
##  [81] -0.840925631 -0.948625155  0.103441447 -0.036129156  0.037037716
##  [86]  0.419072133 -0.449220667 -0.054659265  0.034423764 -0.407061894
##  [91] -0.174138487 -0.679758520 -0.584973289  0.039702057 -0.513246175
##  [96] -0.008488906  0.122011424  0.081831990 -0.019845088  0.900398736
```

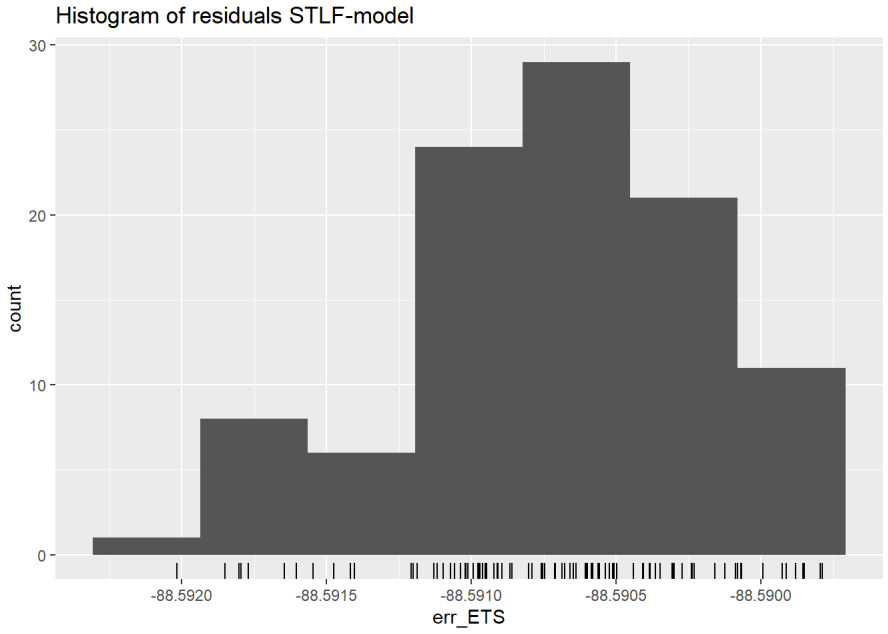
```
summary(mSTLF)
```

```
##
## Forecast method: STL +  ETS(A,N,N)
##
## Model Information:
## ETS(A,N,N)
##
## Call:
##  ets(y = na.interp(x), model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = 0.0076
##
##   sigma:  0.488
##
##      AIC     AICc      BIC
## 55517.69 55517.70 55538.43
##
## Error measures:
##                      ME      RMSE       MAE MPE MAPE     MASE      ACF1
## Training set 0.0012439 0.4879356 0.2292919 NaN  Inf 0.648897 0.0791944
##
## Forecasts:
##           Point Forecast       Lo 80         Hi 80       Lo 95      Hi 95
## 2023.3479   0.0078987364 -0.61750011  0.6332975805 -0.94856626 0.96436373
## 2023.3507   0.3235606824 -0.30183816  0.9489595297 -0.63290432 1.28002568
## 2023.3534   0.0883006686 -0.53709818  0.7136995190 -0.86816434 1.04476567
## 2023.3562   0.1188019823 -0.50659687  0.7442008359 -0.83766303 1.07526699
## 2023.3589  -0.2063420554 -0.83174091  0.4190568013 -1.16280707 0.75012296
## 2023.3616  -0.0043099071 -0.62970877  0.6210889527 -0.96077493 0.95215511
## 2023.3644   0.0808633146 -0.54453555  0.7062621776 -0.87560171 1.03732834
## 2023.3671  -0.0823838915 -0.70778276  0.5430149746 -1.03884892 0.87408114
## 2023.3699  -0.1392099829 -0.76460885  0.4861888863 -1.09567502 0.81725505
## 2023.3726  -0.0141784069 -0.63957728  0.6112204654 -0.97064344 0.94228663
## 2023.3753  -0.0568335305 -0.68223241  0.5685653450 -1.01329857 0.89963151
## 2023.3781   0.2881598754 -0.33723900  0.9135587541 -0.66830517 1.24462492
## 2023.3808  -0.2141666363 -0.83956552  0.4112322455 -1.17063169 0.74229842
## 2023.3836  -0.0774541454 -0.70285303  0.5479447395 -1.03391920 0.87901091
## 2023.3863  -0.0664125681 -0.69181146  0.5589863199 -1.02287763 0.89005249
## 2023.3890  -0.1554891617 -0.78088805  0.4699097294 -1.11195423 0.80097590
## 2023.3918  -0.0179567630 -0.64335566  0.6074421314 -0.97442183 0.93850831
## 2023.3945  -0.0512444480 -0.67664335  0.5741544495 -1.00770952 0.90522063
## 2023.3973  -0.0916977135 -0.71709661  0.5337011871 -1.04816279 0.86476737
## 2023.4000   0.5785646171 -0.04683429  1.2039635208 -0.37790047 1.53502970
## 2023.4027   0.0310263701 -0.59437254  0.6564252769 -0.92543872 0.98749146
## 2023.4055   0.0807742032 -0.54462471  0.7061731132 -0.87569089 1.03723930
## 2023.4082  -0.1236461598 -0.74904507  0.5017527533 -1.08011126 0.83281894
## 2023.4110   0.1771365643 -0.44826235  0.8025354805 -0.77932854 1.13360167
## 2023.4137   0.4167860405 -0.20861288  1.0421849599 -0.53967907 1.37325115
## 2023.4164   0.2263427413 -0.39905618  0.8517416637 -0.73012237 1.18280786
## 2023.4192   0.5646962835 -0.06070264  1.1900952092 -0.39176884 1.52116140
## 2023.4219  -0.2655144883 -0.89091342  0.3598844404 -1.22197961 0.69095064
## 2023.4247  -0.0851297569 -0.71052869  0.5402691750 -1.04159489 0.87133537
## 2023.4274  -0.2089179235 -0.83431686  0.4164810116 -1.16538306 0.74754721
## 2023.4301  -0.0071407973 -0.63253974  0.6182581409 -0.96360594 0.94932434
## 2023.4329  -0.1044256766 -0.72982462  0.5209732647 -1.06089082 0.85203947
## 2023.4356   0.2131625809 -0.41223636  0.8385615253 -0.74330257 1.16962773
## 2023.4384   0.2764056283 -0.34899332  0.9018045758 -0.68005952 1.23287078
## 2023.4411   0.2156495337 -0.40974942  0.8410484844 -0.74081562 1.17211469
## 2023.4438  -0.1178594765 -0.74325843  0.5075394774 -1.07432464 0.83860569
## 2023.4466  -0.0039432203 -0.62934218  0.6214557367 -0.96040839 0.95252195
## 2023.4493  -0.0040343940 -0.62943335  0.6213645661 -0.96049957 0.95243078
## 2023.4521  -0.0041255677 -0.62952453  0.6212733955 -0.96059074 0.95233961
## 2023.4548  -0.0379750194 -0.66337399  0.5874239470 -0.99444020 0.91849016
## 2023.4575   0.0385324979 -0.58686647  0.6639314674 -0.91793269 0.99499768
## 2023.4603   0.2416688974 -0.38373008  0.8670678700 -0.71479629 1.19813409
## 2023.4630  -0.0258756776 -0.65127465  0.5995232981 -0.98234087 0.93058952
## 2023.4658  -0.0945775908 -0.71997657  0.5308213880 -1.05104279 0.86188761
## 2023.4685  -0.2183784698 -0.84377745  0.4070205122 -1.17484367 0.73808674
## 2023.4712  -0.2950763221 -0.92047531  0.3303226630 -1.25154153 0.66138889
## 2023.4740  -0.1571551616 -0.78255415  0.4682438266 -1.11362038 0.79931005
## 2023.4767  -0.2190886245 -0.84448762  0.4063103669 -1.17555384 0.73737659
## 2023.4795  -0.2324675913 -0.85786659  0.3929314033 -1.18893282 0.72399763
## 2023.4822   0.5153214971 -0.11007750  1.1407204947 -0.44114373 1.47178673
## 2023.4849  -0.0812171040 -0.70661610  0.5441818968 -1.03768234 0.87524813
## 2023.4877   0.0886920214 -0.53670698  0.7140910253 -0.86777322 1.04515726
## 2023.4904  -0.0659535389 -0.69135255  0.5594454682 -1.02241878 0.89051170
## 2023.4932   0.0650019758 -0.56039703  0.6904009860 -0.89146327 1.02146722
## 2023.4959   0.1664970618 -0.45890195  0.7918960751 -0.78996819 1.12296231
## 2023.4986  -0.1643191572 -0.78971817  0.4610798593 -1.12078441 0.79214610
## 2023.5014   0.1891795971 -0.43621942  0.8145786167 -0.76728567 1.14564486
## 2023.5041   0.2867572056 -0.33864182  0.9121562284 -0.66970806 1.24322247
```

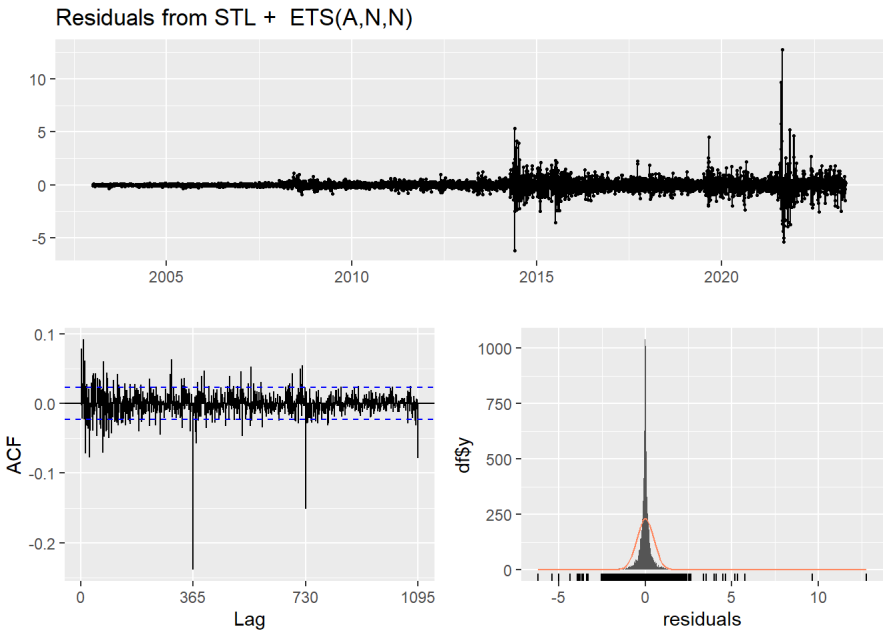
```
autoplot(ts(err_STLF)) + xlab("Day") + ylab("") +
  ggtitle("Residuals from STLF-model")
```




```
gghistogram(err_ETS) + ggtitle("Histogram of residuals STLF-model")
```




```
checkresiduals(mSTLF)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from STL +  ETS(A,N,N)
## Q* = 2566.2, df = 730, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 730
```

В результате наименьшее MASE у моделей ETS MASE = 0.03547228, STLF MASE= 0.03711753, ARIMA(1,1,1) MASE = 0.620264, Сравним эти модели с помощью

Diebold-Mariano test

```
dm.test(err_AUTO.ARIMA, err_ARIMA2,
        alternative = "two.sided",
        h = 1, power = 1)
```

```
##
##  Diebold-Mariano Test
##
## data:  err_AUTO.ARIMAerr_ARIMA2
## DM = 208.05, Forecast horizon = 1, Loss function power = 1, p-value <
## 2.2e-16
## alternative hypothesis: two.sided
```

```
dm.test(err_AUTO.ARIMA, err_ETS,
        alternative = "two.sided",
        h = 1, power = 1)
```

```
##
##  Diebold-Mariano Test
##
## data:  err_AUTO.ARIMAerr_ETS
## DM = -80.269, Forecast horizon = 1, Loss function power = 1, p-value <
## 2.2e-16
## alternative hypothesis: two.sided
```

```
dm.test(err_AUTO.ARIMA, err_STLF,
        alternative = "two.sided",
        h = 1, power = 1)
```

```
##
##  Diebold-Mariano Test
##
## data:  err_AUTO.ARIMAerr_STLF
## DM = 205.17, Forecast horizon = 1, Loss function power = 1, p-value <
## 2.2e-16
## alternative hypothesis: two.sided
```

```
dm.test(err_ARIMA2, err_ETS,
        alternative = "two.sided",
        h = 1, power = 1)
```

```
##
##  Diebold-Mariano Test
##
## data:  err_ARIMA2err_ETS
## DM = -21773, Forecast horizon = 1, Loss function power = 1, p-value <
## 2.2e-16
## alternative hypothesis: two.sided
```

```
dm.test(err_ARIMA2, err_STLF,
        alternative = "two.sided",
        h = 1, power = 1)
```

```
##
##  Diebold-Mariano Test
##
## data:  err_ARIMA2err_STLF
## DM = -7.8972, Forecast horizon = 1, Loss function power = 1, p-value =
## 3.983e-12
## alternative hypothesis: two.sided
```

```
dm.test(err_ETS, err_STLF,
        alternative = "two.sided",
        h = 1, power = 1)
```

```
##
##  Diebold-Mariano Test
##
## data:  err_ETSerr_STLF
## DM = 1677.7, Forecast horizon = 1, Loss function power = 1, p-value <
## 2.2e-16
## alternative hypothesis: two.sided
```

Результаты MASE:

ETS - (Errors, Trend, Seasonal) (без учета первых разниц) MASE = 0.1093651 MASE = 0.03547228 (summary)

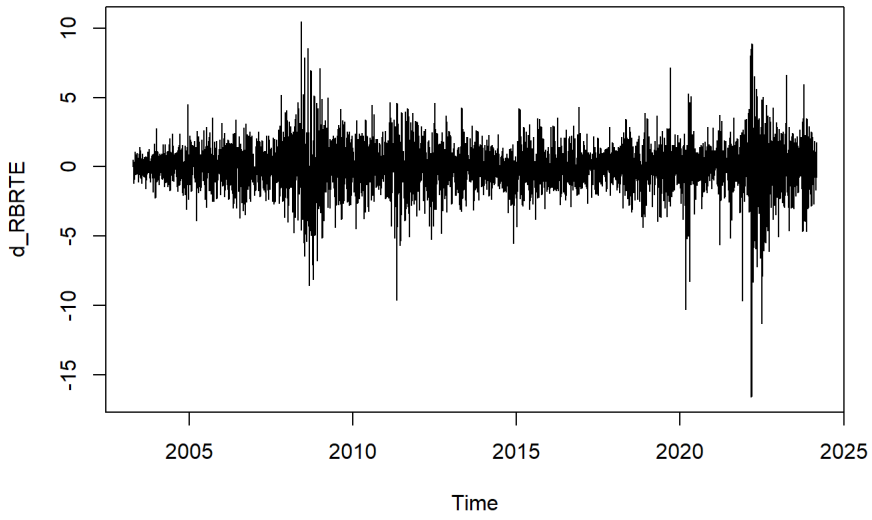
ETS1 - model (Errors, Trend, Seasonal) (with first differences) (с первыми разницами) MASE = 0.612617 MASE = 1.479459 (summary)

STLF (Seasonal and Trend decomposition using Loess) - метод основан на локальной регрессии (Loess) MASE = 0.0415422 MASE = 0.03711753 (summary)

Структурные сдвиги

Используем цены на нефть в качестве предиктора

```
tsRBRTE=ts(dd$RBRTE,
       start = c(2003,104),
       frequency = 365)

d_RBRTE = diff(tsRBRTE)

plot(d_RBRTE)
```




```
trainRBRTE = tsRBRTE[1:(length(tsRBRTE) - nToPred - 1)]
trainRBRTE=ts(trainRBRTE,
       start = c(2003,104),
       frequency = 365)
d_trainRBRTE = d_RBRTE[1:(length(d_RBRTE) - nToPred - 1)]
d_trainRBRTE=ts(d_trainRBRTE,
       start = c(2003,104),
       frequency = 365)
```

```
# Chow test based breaks
#Fres = strucchange::Fstats(trainUSD ~ trainRBRTE)
#plot(Fres) # показывает где есть сдвиги и какие большие
```

```
# Chow test based breaks
#d_Fres = strucchange::Fstats(d_trainUSD ~ d_trainRBRTE)
#plot(d_Fres) # показывает где есть сдвиги и какие большие
```

В то время как обобщение теста Чоу не выявляет значимых сдвигов для общего ряда, сдвиги наблюдаются в первых 2/3 от временного ряда разности.

```
#Fres$breakpoint
#Fres$datatsp

#d_Fres$breakpoint
#d_Fres$datatsp
```

Точка разлома находятся в 24 июл. 2014 г.- 16 авг. 2014 г. основываясь на двух рядах.

```
# CUSUM type

#CUSUMres = efp(trainUSD ~ trainRBRTE)
#plot(CUSUMres) # структурные сдвиги с
```

```
# CUSUM type

#d_CUSUMres = efp(d_trainUSD ~ d_trainRBRTE)
#plot(d_CUSUMres)
```

```
# MOSUM

#MOSUMres = efp(trainUSD ~ trainRBRTE, type = "Rec-MOSUM")
#plot(MOSUMres)

#?efp()
```

Структурные сдвиги с 2008, увеличиваются с 2014 по 2016-2017 и дальше с середины 2020 - то есть практически во все теоретические кризисные для России и США моменты

```
# MOSUM

#d_MOSUMres = efp(d_trainUSD ~ d_trainRBRTE, type = "Rec-MOSUM")
#plot(d_MOSUMres)
```

```
# Bai-Perron

#BRres = breakpoints(trainUSD ~ trainRBRTE)
#plot(BRres) # оптимально по БИКу  сдвига
#summary(BRres) # количество наблюдений с разными сдвигами, даты
```

В итоге, можно было бы прогнать нашу лучшую модель на урезанных обучающих данных с 2024г, например, но у нас прогоняется всё по 3 часа :(