Graduation Project

Mustafa Nayansak - 528191047

12-07-2020

Contents

Introduction

In this report, Value at Risk research was conducted for Banvit and Net holding stocks. The data set consists of approximately 260000 lines of observations and the data is in milliseconds. In the data set, operations such as filling missing data, combining stock data, and time synchronization were performed. In addition, 4 different data sets were created for 1 minute, 15 minutes, 30 minutes and daily, and the effects of the methods and significance levels used on the data set were investigated.

In the study, the value at risk research was investigated in four different ways: parametric, historical, monte carlo and age weighted. Additionally, the performance of the related value at risk method was measured using the sliding method. Finally, statistical analysis of exceedence was done using backtesting methods.

Data

Net Holding (nthol)

```
readxl::read_xlsx("stocks202005111746b.xlsx", sheet = "nthol") %>%
  arrange(Timestamp) %>%
  mutate(symbol = "nthol",
         Timestamp = format(as.POSIXct(Timestamp),
                            format = "%Y-%m-%d %H:%M:%OS3")) %>%
  head() %>% kable()
```

Timestamp	Last	Trade Volume	symbol
2020-02-11 17:44:30.999	2.35	750	nthol
2020-02-11 17:44:34.381	2.36	102	nthol
2020-02-11 17:44:42.312	2.36	2830	nthol
2020-02-11 17:46:14.473	2.36	5000	nthol
2020-02-11 17:47:08.355	2.36	1206	nthol
2020-02-11 17:47:29.885	2.36	14841	nthol

Data Summary

Descriptive statistics are very informative about the distribution of a data set. As can be seen from the table, although there is not much difference between the minimum and maximum values of the closing values of the nthol stock, their volumes have distant values.

```
nthol %>% select_if(is.numeric) %>% summary() %>% kable()
```

	Last	Trade Volume
	Min. :1.110	Min. : 1
	1st Qu.:1.500	1st Qu.: 5
	Median :1.870	Median : 161
	Mean :1.869	Mean : 2807
	3rd Qu.:2.110	3rd Qu.: 2000
	Max. :2.720	Max. :287645
	NA's :5271	NA's :5271

Banvit (banvt)

The time difference of the stock values of banvit and net holding is about 7 minutes. Therefore, these data sets should be combined with various manipulation procedures.

```
readxl::read_xlsx("stocks202005111746b.xlsx", sheet = "banvt") %>%
  arrange(Timestamp) %>%
  mutate(symbol = "banvt",
         Timestamp = format(as.POSIXct(Timestamp),
           format = "%Y-%m-%d %H:%M:%OS3")) %>%
  head() %>% kable()
```

Timestamp	Last	Trade Volume	symbol
2020-02-11 17:36:34.999	18.87	20	banvt
2020-02-11 17:36:34.999	18.86	1	banvt
2020-02-11 17:40:53.486	18.88	170	banvt
2020-02-11 17:40:53.486	18.87	350	banvt
2020-02-11 17:40:53.486	18.87	230	banvt
2020-02-11 17:42:25.950	18.87	9	banvt

Data Summary

```
banvt %>% select_if(is.numeric) %>% summary() %>% kable()
```

	Last	Trade Volume
	Min. :13.14	Min. : 1.0
	1st Qu.:17.70	1st Qu.: 10.0
	Median :19.99	Median : 71.0
	Mean :21.60	Mean : 231.9
	3rd Qu.:26.26	3rd Qu.: 250.0
	Max. :31.64	Max. :31410.0
	NA's :6447	NA's :6447

Data Manipulation

Creating Different Time Periods

Below are three different ways to separate two stock data in minutes, 15 minutes and 30 minutes. In addition, opening, closing, minimum and maximum values of the relevant time interval were calculated. A new variable was created by taking the average of the minimum and maximum values obtained here. In this way, it is aimed to take into account the variability in the data.

```
banvt_1m <- cbind(to.minutes(banvt_xts),rowMeans(to.minutes(banvt_xts)[,2:3]))[,5]
nthol_1m <- cbind(to.minutes(nthol_xts),rowMeans(to.minutes(nthol_xts)[,2:3]))[,5]

banvt_15m <- cbind(to.minutes15(banvt_xts),rowMeans(to.minutes15(banvt_xts)[,2:3]))[,5]
nthol_15m <- cbind(to.minutes15(nthol_xts),rowMeans(to.minutes15(nthol_xts)[,2:3]))[,5]

banvt_30m <- cbind(to.minutes30(banvt_xts),rowMeans(to.minutes30(banvt_xts)[,2:3]))[,5]
nthol_30m <- cbind(to.minutes30(nthol_xts),rowMeans(to.minutes30(nthol_xts)[,2:3]))[,5]
```

Time Synchronization

Considering that after combining the datasets, it would not be possible to operate on both datasets at each time point, it was necessary to merge the two datasets with all their values. After this process, the data set was created by filling the missing values according to the full observation values above. Finally, with the ROC() function, returns are calculated and added to the data sets.

```
# MERGING XTS FILES

  ## 1M
default_data_1 <- merge.xts(banvt = banvt_1m, nthol = nthol_1m, all = TRUE)
xts_wo_1 <- na.locf(default_data_1, fromLast=TRUE) %>% na.exclude() %>%
  `colnames<-`(c("banvt", "nthol"))
tbl_1 <- xts_wo_1 %>% fortify.zoo %>% as_tibble() %>% rename("Timestamp" = "Index")

    ### 1M - RETURNS
tbl_1 %<>% mutate(banvt_return = ROC(banvt),
                  nthol_return = ROC(nthol)) %>% na.exclude()

  ## 15M
default_data_15 <- merge.xts(banvt = banvt_15m, nthol = nthol_15m, all = TRUE)
xts_wo_15 <- na.locf(default_data_15, fromLast=TRUE) %>% na.exclude() %>%
  `colnames<-`(c("banvt", "nthol"))
tbl_15 <- xts_wo_15 %>% fortify.zoo %>% as_tibble() %>% rename("Timestamp" = "Index")

    ### 15M - RETURNS
tbl_15 %<>% mutate(banvt_return = ROC(banvt),
                   nthol_return = ROC(nthol)) %>% na.exclude()

  ## 30M
default_data_30 <- merge.xts(banvt = banvt_30m, nthol = nthol_30m, all = TRUE)
xts_wo_30 <- na.locf(default_data_30, fromLast=TRUE) %>% na.exclude() %>%
  `colnames<-`(c("banvt", "nthol"))
tbl_30 <- xts_wo_30 %>% fortify.zoo %>% as_tibble() %>% rename("Timestamp" = "Index")

    ### 30M - RETURNS
tbl_30 %<>% mutate(banvt_return = ROC(banvt),
                   nthol_return = ROC(nthol)) %>% na.exclude()
```

30 Minutes Data

The final data set obtained after the manipulation procedures applied is as follows.

```
tbl_30 %>% head() %>% kable()
```

Timestamp	banvt	nthol	banvt_return	nthol_return
2020-02-11 17:59:59	18.96	2.360	0.0055534	0.0000000
2020-02-11 18:08:00	18.96	2.360	0.0000000	0.0000000
2020-02-11 18:08:50	18.86	2.360	-0.0052882	0.0000000
2020-02-12 10:25:58	18.86	2.355	0.0000000	-0.0021209
2020-02-12 10:28:58	18.81	2.355	-0.0026546	0.0000000
2020-02-12 10:57:52	18.81	2.370	0.0000000	0.0063492

Understanding Value at Risk (VaR)

Value at risk (VaR) is a statistic that measures and quantifies the level of financial risk within a firm, portfolio or position over a specific time frame. This metric is most commonly used by investment and commercial banks to determine the extent and occurrence ratio of potential losses in their institutional portfolios.

To understand the logic of Value at risk, firstly, parametric, historical, monte carlo and age weighted historical simulation values were calculated according to a single time interval (30M) and a single significance value (0.05).

Parametric Value at Risk

Under the assumption that the returns are distributed normally, the following calculation is done using the average and standard deviations of the returns. Value at Risk values of two stocks are as follows.

```
mean_dr_banvt <- mean(tbl_30 %>% pull(banvt_return))
mean_dr_nthol <- mean(tbl_30 %>% pull(nthol_return))

sd_dr_banvt <- sd(tbl_30 %>% pull(banvt_return))
sd_dr_nthol <- sd(tbl_30 %>% pull(nthol_return))

parametric_var_banvt <- -mean_dr_banvt + z*sd_dr_banvt
parametric_var_nthol <- -mean_dr_nthol + z*sd_dr_nthol
tibble(Parametric_banvt = parametric_var_banvt,
       Parametric_nthol = parametric_var_nthol) %>% kable()
```

Parametric_banvt	Parametric_nthol
-0.0152989	-0.0125622

Historical Value at Risk

Calculated using quantiles in the historical VAR dataset. In this section, VaR values corresponding to 0.05 quantile values calculated at 0.05 significance level are as follows.

```
historical_var_banvt <- quantile(tbl_30 %>% pull(banvt_return), alpha_)
historical_var_nthol <- quantile(tbl_30 %>% pull(nthol_return), alpha_)
tibble(historical_banvt = historical_var_banvt,
       historical_nthol = historical_var_nthol) %>% kable()
```

historical_banvt	historical_nthol
-0.0044688	-0.0049286

Monte Carlo Simulation

Monte carlo simulation is known as a very effective method in cases where the distribution of a data set is known. In this section, a Monte Carlo simulation, which is repeated 1000 times, is created under the assumption that returns are distributed normally.

```
mean_dr_banvt <- tbl_30 %>% pull(banvt_return) %>% mean()
mean_dr_nthol <- tbl_30 %>% pull(nthol_return) %>% mean()

calculate_one_period_change <- function(mean, sd){-mean+sd*rnorm(1)}
simulated_returns_banvt <- replicate(1000,
                                     calculate_one_period_change(mean = mean_dr_banvt,
                                                                 sd = sd_dr_banvt))
simulated_returns_nthol <- replicate(1000,
                                         calculate_one_period_change(mean = mean_dr_nthol,
                                                                     sd = sd_dr_nthol))

mc_var_banvt <- quantile(simulated_returns_banvt, alpha_)
mc_var_nthol <- quantile(simulated_returns_nthol, alpha_)

tibble(mc_banvt = mc_var_banvt,
       mc_nthol = mc_var_nthol) %>% kable()
```

mc_banvt	mc_nthol
-0.0116601	-0.0093344

Age-Weighted Historical Simulation

The Age wieghted historical simulation method calculates VaR by giving more weight to recent occurrences rather than giving equal weights to each observation value. Here the weighting parameter is the lambda parameter.

```
lambda <- 0.98

basic_histor_banvt <- tbl_30 %>% select(banvt, banvt_return) %>%
  mutate(Periods_30 = 1:nrow(tbl_30)) %>%
  arrange(banvt_return) %>%
  mutate(eq_weights = 1/nrow(tbl_30),
         cum_eq_weigths = cumsum(eq_weights))

ewh_banvt <- basic_histor_banvt %>%
  mutate(hybrid_weights = ((1-lambda)*(lambda^(Periods_30-1)))/(1-lambda^k),
         hybrid_cum_weights = cumsum(hybrid_weights))

ewh_var_banvt <- approx(x =ewh_banvt$hybrid_cum_weights,
       y = ewh_banvt$banvt_return,
       xout=alpha_)$y
```

```
basic_histor_nthol <- tbl_30 %>% select(nthol, nthol_return) %>%
  mutate(Periods_30 = 1:nrow(tbl_30)) %>%
  arrange(nthol_return) %>%
  mutate(eq_weights = 1/nrow(tbl_30),
         cum_eq_weigths = cumsum(eq_weights))

ewh_nthol <- basic_histor_nthol %>%
  mutate(hybrid_weights = ((1-lambda)*(lambda^(Periods_30-1)))/(1-lambda^k),
         hybrid_cum_weights = cumsum(hybrid_weights))

ewh_var_nthol <- approx(x =ewh_nthol$hybrid_cum_weights,
       y = ewh_nthol$nthol_return,
       xout=alpha_)$y

tibble(ewh_banvt = ewh_var_banvt,
       ewh_nthol = ewh_var_nthol) %>% kable()
```

ewh_banvt	ewh_nthol
-0.0071485	-0.0042415

Overall

VaR values calculated with different methods at the significance level of 0.05 are listed as follows. Backtesting methods will be used in the following sections to test the accuracy of these values.

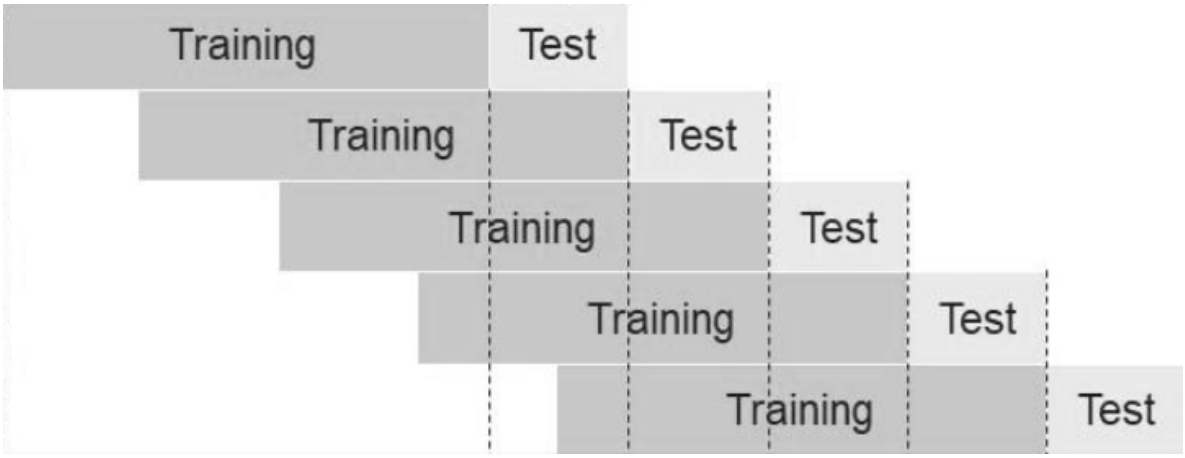
```
tibble("VaR" = c("Parametric", "Historical", "Monte Carlo", "Age Weighted HS"),
       "Bant" = c(parametric_var_banvt, historical_var_banvt, mc_var_banvt, ewh_var_banvt),
       "Nthol" = c(parametric_var_nthol, historical_var_nthol, mc_var_nthol, ewh_var_nthol)) %>%
  kable()
```

VaR	Bant	Nthol
Parametric	-0.0152989	-0.0125622
Historical	-0.0044688	-0.0049286
Monte Carlo	-0.0116601	-0.0093344
Age Weighted HS	-0.0071485	-0.0042415

Applied Value at Risk (Sliding)

It will be more accurate and consistent to test the accuracy of value at risk values with sliding method. As it is seen in the figure, a structure was established that takes 252 observations as training and progresses by predicting the next observation (observation 253).

```
img <- readPNG("C:/Users/mnayansak/Desktop/sliding.PNG")
grid.raster(img)
```



alpha = 0.05 | 30 Minutes

Sliding processes are applied on different VaR calculation techniques and different data sets. Firstly, Value at Risk values were calculated with a 30 minute banvit data set at 0.05 significance level. According to the calculated values in each sliding operation, a variable encoded as 0 in case of any excess and 1 in case of absence is added.

Banvit | Average Performance

The success performances of the methods used are calculated by taking the average of the variable consisting of 0 and 1 values indicating whether there are excesses. According to the table, the best result was found to be Monte Carlo method.

```
res_CM_bv_3095 %>%
  kable()
```

	x
parametric_VaR	0.9696468
historic_VaR	0.9459161
monte_carlo_VaR	0.9735099
awh_VaR	0.9387417

Backtests

Kupiec and Christoffesen backtest methods have been used to check whether the exceedence are statistically significant and systematic.

Kupiec Tests (POF)

$H_0 : p = \hat{p}$

$H_A : p \neq \hat{p}$

Christoffesen Test

H_0 : Independent failures

H_A : Dependent failures

```
res_BT_bv_3095 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_banvt	1.0e-06	0.4308775	5.0e-07	0.0333916
Christoffesen_banvt	5.3e-06	0.7260897	2.7e-06	0.0984113

According to the test results, it was concluded that the values calculated with historical VaR (1-alpha) were statistically compatible with the confidence level and exceedence were not systematic.

Net Holding | Average Performance

In the 30-minute netholding data, the Parametric VaR and Monte Carlo VaR methods with a significance level of 0.05 worked with better performance.

```
res_CM_nt_3095 %>%
  kable()
```

	x
parametric_VaR	0.9707506
historic_VaR	0.9431567
monte_carlo_VaR	0.9701987
awh_VaR	0.9420530

Backtests

It is possible to say that historical VaR and age weighted HS VaR methods are more positive than backtest methods.

```
res_BT_nt_3095 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_Nthol	1.19e-05	0.1904489	2.11e-05	0.1296170
Christoffesen_Nthol	3.86e-05	0.2936308	7.07e-05	0.3168713

alpha = 0.05 | 15 Minutes

Banvit | Average Performance

In the 15-minute banvit data, the Parametric VaR and Monte Carlo VaR methods with a significance level of 0.05 worked with better performance.

```
res_CM_bv_1595 %>%
  kable()
```

	x
parametric_VaR	0.9738108
historic_VaR	0.9428113
monte_carlo_VaR	0.9738108
awh_VaR	0.9358632

Backtests

Kupiec Tests (POF)

$H_0 : p = \hat{p}$

$H_A : p \neq \hat{p}$

Christoffesen Test

H_0 : independent failures

H_A : dependent failures

```
res_BT_bv_1595 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_banvt	0	0.0483293	0	0.0000440
Christoffesen_banvt	0	0.0000517	0	0.0124996

As the frequency of the data increases, the results of VaR values have been inconsistent. For this reason, daily calculations are made in the last section.

Net Holding | Average Performance

In the 15-minute nthol data, the Parametric VaR and Monte Carlo VaR methods with a significance level of 0.05 worked with better performance.

```
res_CM_nt_1595 %>%
  kable()
```

	x
parametric_VaR	0.9724746
historic_VaR	0.9361304
monte_carlo_VaR	0.9727418
awh_VaR	0.9404062

Backtests

```
res_BT_nt_1595 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_Nthol	0	0.0002423	0	0.0407278
Christoffesen_Nthol	0	0.0000517	0	0.0124996

As the frequency increases as above, the results were observed to be inconsistent.

alpha = 0.05 | 1 Minute

Banvit | Average Performance

In the 1-minute banvit data, the Parametric VaR and Monte Carlo VaR methods with a significance level of 0.05 worked with better performance.

```
res_CM_bv_195 %>%
  kable()
```

	x
parametric_VaR	0.9772098
historic_VaR	0.9533681
monte_carlo_VaR	0.9770381
awh_VaR	0.9426597

Backtests

Kupiec Tests (POF)

$H_0 : p = \hat{p}$

$H_A : p \neq \hat{p}$

Christoffesen Test

H_0 : independent failures

H_A : dependent failures

```
res_BT_bv_195 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec	0	0.0007448	0	0
Christoffesen	0	0.0000000	0	0

P-values decreased as frequency increased. And hypotheses tend to be strongly rejected in this way.

Net Holding | Average Performance

Unlike others, nthol data at 1 minute frequency works better with Age Weighted HS.

```
res_CM_nt_195 %>%
  kable()
```

	x
parametric_VaR	0.9583467
historic_VaR	0.9386253
monte_carlo_VaR	0.9573381
awh_VaR	0.9781111

Backtests

```
res_BT_nt_195 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_Nthol	0	0	0	0
Christoffesen_Nthol	0	0	0	0

VaR Performance | alpha = 0.05

In the generalized table below, it is possible to say which method works better in which data set. As a result, it is possible to say that the Monte Carlo simulation method shows an average of around 97% performance in all data sets.

```
data.frame(banvt_30_95 %>% colMeans(na.rm = T),
           banvt_15_95 %>% colMeans(na.rm = T),
           banvt_1_95 %>% colMeans(na.rm = T),
           nthol_30_95 %>% colMeans(na.rm = T),
           nthol_15_95 %>% colMeans(na.rm = T),
           nthol_1_95 %>% colMeans(na.rm = T)) %>% `colnames<-`(c("Min=30|banvt",
                                                                  "Min=15|banvt",
                                                                  "Min=1|banvt",
                                                                  "Min=30|nthol",
                                                                  "Min=15|nthol",
                                                                  "Min=1|nthol")) %>%
  kable()
```

	Min=30|banvt	Min=15|banvt	Min=1|banvt	Min=30|nthol	Min=15|nthol	Min=1|nthol
parametric_VaR	0.9696468	0.9738108	0.9772098	0.9707506	0.9724746	0.9583467
historic_VaR	0.9459161	0.9428113	0.9533681	0.9431567	0.9361304	0.9386253
monte_carlo_VaR	0.9735099	0.9738108	0.9770381	0.9701987	0.9727418	0.9573381
awh_VaR	0.9387417	0.9358632	0.9426597	0.9420530	0.9404062	0.9781111

alpha = 0.01 | 30 Minutes

Banvit | Average Performance

As the Alpha value decreases, the risk decreases even more. Consequently, VaR results will be even lower. It is possible to see that the average results obtained below increase.

```
res_CM_bv_3099 %>%
  kable()
```

	x
parametric_VaR	0.9729581
historic_VaR	0.9895143
monte_carlo_VaR	0.9862031
awh_VaR	0.9752169

Backtests

Kupiec Tests (POF)

$H_0 : p = \hat{p}$

$H_A : p \neq \hat{p}$

Christoffesen Test

H_0 : independent failures

H_A : dependent failures

According to the calculated backtest results, it was concluded that exceedence of Historical VaR and Monte Carlo VaR values were statistically compatible with 0.99 and exceedence were not systematic.

```
res_BT_bv_3099 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_banvt	0	0.8291296	0.1061241	0.0000005
Christoffesen_banvt	0	0.1696490	0.0843325	0.0553245

Net Holding | Average Performance

It is observed that Historical VaR and Monte Carlo VaR methods work with higher performance in nthol (30M) data.

```
res_CM_nt_3099 %>%
  kable()
```

	x
parametric_VaR	0.9707506
historic_VaR	0.9850993
monte_carlo_VaR	0.9834437
awh_VaR	0.9739583

Backtests

According to the backtest methods, the historical VaR method is more consistent than the other methods.

```
res_BT_nt_3099 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_Nthol	0	0.1091883	0.0403675	0.0232541
Christoffesen_Nthol	0	0.1696490	0.0843325	0.0553245

alpha = 0.01 | 15 Minutes

Banvit | Average Performance

It is observed that Historical VaR and Monte Carlo VaR methods work with higher performance in banvt(15M) data.

```
res_CM_bv_1599 %>%
  kable()
```

	x
parametric_VaR	0.9738108
historic_VaR	0.9874399
monte_carlo_VaR	0.9863709
awh_VaR	0.9776211

Backtests

Kupiec Tests (POF)

$H_0 : p = \hat{p}$

$H_A : p \neq \hat{p}$

Christoffesen Test

H_0 : independent failures

H_A : dependent failures

```
res_BT_bv_1599 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_banvt	0	0.0812634	0.0264605	0
Christoffesen_banvt	0	0.0000963	0.0002076	0

As in the previous sections, the significance of the methods disappears as the frequency increases.

Net Holding | Average Performance

It is observed that Historical VaR and Monte Carlo VaR methods work with higher performance in nthol (15M) data.

```
res_CM_nt_1599 %>%
  kable()
```

	x
parametric_VaR	0.9724746
historic_VaR	0.9855692
monte_carlo_VaR	0.9874399
awh_VaR	0.9760795

Backtests

```
res_BT_nt_1599 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_Nthol	0	0.0089898	0.0965543	0
Christoffesen_Nthol	0	0.0000963	0.0002076	0

It is possible to say for this section that the results are meaningless as the frequency increases.

alpha = 0.01 | 1 Minute

Banvit | Average Performance

It is observed that Historical VaR and Monte Carlo VaR methods work with higher performance in banvy (1M) data.

```
res_CM_bv_199 %>%
  kable()
```

	x
parametric_VaR	0.9772098
historic_VaR	0.9993133
monte_carlo_VaR	0.9879397
awh_VaR	0.9752731

Backtests

Kupiec Tests (POF)

$H_0 : p = \hat{p}$

$H_A : p \neq \hat{p}$

Christoffesen Test

H_0 : independent failures

H_A : dependent failures

```
res_BT_bv_199 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_banvt	0	0	0	0
Christoffesen_banvt	0	0	0	0

Net Holding | Average Performance

It can be said that Historical VaR and Age Weighted HS methods are over 99% successful according to VaR values calculated at a significance level of 0.01 minutes. However, it is not the right way to work at this frequency since the backtest results will be meaningless due to the high frequency.

```
res_CM_nt_199 %>%
  kable()
```

	x
parametric_VaR	0.9583467
historic_VaR	0.9992918
monte_carlo_VaR	0.9757935
awh_VaR	0.9926180

Backtests

```
res_BT_nt_199 %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_Nthol	0	0	0	0
Christoffesen_Nthol	0	0	0	0

VaR Performance | alpha = 0.01

It is observed that when the Alpha value decreases, the historical VaR method gives more consistent results when applied to all data sets.

```
data.frame(banvt_30_99 %>% colMeans(na.rm = T),
           banvt_15_99 %>% colMeans(na.rm = T),
           banvt_1_99 %>% colMeans(na.rm = T),
           nthol_30_99 %>% colMeans(na.rm = T),
           nthol_15_99 %>% colMeans(na.rm = T),
           nthol_1_99 %>% colMeans(na.rm = T)) %>% `colnames<-`(c("Min=30|banvt",
                                                                  "Min=15|banvt",
                                                                  "Min=1|banvt",
                                                                  "Min=30|nthol",
                                                                  "Min=15|nthol",
                                                                  "Min=1|nthol")) %>%
  kable()
```

	Min=30|banvt	Min=15|banvt	Min=1|banvt	Min=30|nthol	Min=15|nthol	Min=1|nthol
parametric_VaR	0.9729581	0.9738108	0.9772098	0.9707506	0.9724746	0.9583467
historic_VaR	0.9895143	0.9874399	0.9993133	0.9850993	0.9855692	0.9992918
monte_carlo_VaR	0.9862031	0.9863709	0.9879397	0.9834437	0.9874399	0.9757935
awh_VaR	0.9752169	0.9776211	0.9752731	0.9739583	0.9760795	0.9926180

VaR | Daily Calculation

When the data set is translated daily, the number of observations in the data is 61 and errors occur in calculations with confidence levels such as 99% and 95%. Therefore, alpha value is chosen as 0.10.

The sliding method and VaR calculation methods used in the previous analyzes are applied as follows. Sliding processes were applied in 14 day steps.

```
alpha_ <- 0.1

banvt_daily <- cbind(to.daily(banvt_xts),rowMeans(to.daily(banvt_xts)[,2:3]))[,5]
nthol_daily <- cbind(to.daily(nthol_xts),rowMeans(to.daily(nthol_xts)[,2:3]))[,5]

daily_data <- merge.xts(banvt = banvt_daily, nthol = nthol_daily, all = TRUE)
daily_data_wo <- na.locf(daily_data, fromLast=TRUE) %>% na.exclude() %>%
  `colnames<-`(c("banvt", "nthol"))
tbl <- daily_data_wo %>% fortify.zoo %>% as_tibble() %>% rename("Timestamp" = "Index")

tbl %<>% mutate(banvt_return = ROC(banvt),
                nthol_return = ROC(nthol)) %>% na.exclude()

banvt_d <- data.frame(parametric_VaR = NA,
                      historic_VaR = NA,
                      monte_carlo_VaR = NA,
```

```
                          awh_VaR = NA)
nthol_d <- data.frame(parametric_VaR = NA,
                          historic_VaR = NA,
                          monte_carlo_VaR = NA,
                          awh_VaR = NA)
banvt_d_backtest <- data.frame(Actual = NA, param_VaR = NA, hist_VaR = NA,
                               mc_VaR = NA, ewhs_VaR = NA)
nthol_d_backtest <- data.frame(Actual = NA, param_VaR = NA, hist_VaR = NA,
                                mc_VaR = NA, ewhs_VaR = NA)

k <- 14
for(i in 1:((nrow(tbl)-k)-1)){

  ## PARAMETRIC VaR

  mean_banvt <-  mean(tbl %>% slice(1:(k+i-1)) %>% pull(banvt_return))
  mean_nthol <- mean(tbl %>% slice(1:(k+i-1)) %>% pull(nthol_return))

  sd_dr_banvt <- sd(tbl %>% slice(1:(k+i-1)) %>% pull(banvt_return))
  sd_dr_nthol <- sd(tbl %>% slice(1:(k+i-1)) %>% pull(nthol_return))

  banvt_d_backtest[,1] <- tbl %>% slice((k+1):nrow(tbl)) %>% select(banvt_return)
  banvt_d_backtest[i,2] <- -mean_banvt + z*sd_dr_banvt

  nthol_d_backtest[,1] <- tbl %>% slice((k+1):nrow(tbl)) %>% select(nthol_return)
  nthol_d_backtest[i,2] <- -mean_nthol + z*sd_dr_nthol

  banvt_d[i,1] <- ifelse(-mean_banvt + z*sd_dr_banvt >
                           tbl %>% select(banvt_return) %>%
                         filter(row_number() == (k+i)) %>% pull(),
                          0,1)

  nthol_d[i,1] <- ifelse(-mean_nthol + z*sd_dr_nthol >
                           tbl %>% select(nthol_return) %>%
                         filter(row_number() == (k+i)) %>% pull(),
                          0,1)

  ## HISTORICAL VaR
  banvt_q <- quantile(tbl %>% slice(1:(k+i-1)) %>% pull(banvt_return), alpha_)
  nthol_q <- quantile(tbl %>% slice(1:(k+i-1)) %>% pull(nthol_return), alpha_)

  banvt_d_backtest[i,3] <- banvt_q
  nthol_d_backtest[i,3] <- nthol_q

  banvt_d[i,2] <- ifelse(banvt_q>
                           tbl %>% select(banvt_return) %>%
                         filter(row_number() == (k+i)) %>% pull(),
                          0,1)
  nthol_d[i,2] <- ifelse(nthol_q>
```

```
                              tbl %>% select(nthol_return) %>%
                            filter(row_number() == (k+i)) %>% pull(),
                             0,1)

## MONTE CARLO SIMULATION
mean_dr_banvt <- tbl %>% slice(1:(k+i-1)) %>% pull(banvt_return) %>% mean()
mean_dr_nthol <- tbl %>% slice(1:(k+i-1)) %>% pull(nthol_return) %>% mean()

simulated_returns_banvt <- replicate(1000,
                                     calculate_one_period_change(mean = mean_dr_banvt,
                                                                 sd = sd_dr_banvt))
simulated_returns_nthol <- replicate(1000,
                                     calculate_one_period_change(mean = mean_dr_nthol,
                                                                 sd = sd_dr_nthol))

banvt_d_backtest[i,4] <- quantile(simulated_returns_banvt, alpha_)
nthol_d_backtest[i,4] <- quantile(simulated_returns_nthol, alpha_)

banvt_d[i,3] <- ifelse(quantile(simulated_returns_banvt, alpha_)>
                              tbl %>% select(banvt_return) %>%
                            filter(row_number() == (k+i)) %>% pull(),
                             0,1)

nthol_d[i,3] <- ifelse(quantile(simulated_returns_nthol, alpha_)>
                              tbl %>% select(nthol_return) %>%
                            filter(row_number() == (k+i)) %>% pull(),
                             0,1)

# AGE-WEIGHTED HISTORICAL SIMULATION | INTERPOLATION

## BANVT
basic_histor_banvt <- tbl %>% select(banvt, banvt_return) %>%
  slice((1+i-1):(k+i-1)) %>%
  mutate(Periods_1 = 1:k) %>%
  arrange(banvt_return) %>%
  mutate(eq_weights = 1/k,
         cum_eq_weigths = cumsum(eq_weights))

awh_banvt <- basic_histor_banvt %>%
  mutate(hybrid_weights = ((1-lambda)*(lambda^(Periods_1-1)))/(1-lambda^k),
         hybrid_cum_weights = cumsum(hybrid_weights))

banvt_d_backtest[i,5] <- approx(x =awh_banvt$hybrid_cum_weights,
                                y = awh_banvt$banvt_return,
                                xout=alpha_)$y

banvt_d[i,4] <- ifelse(approx(x = awh_banvt$hybrid_cum_weights,
                              y = awh_banvt$banvt_return,
                              xout=alpha_)$y>
```

```
                          tbl %>% select(banvt_return) %>%
                          filter(row_number() == k+i) %>% pull(),
                        0,1)

  ## NTHOL
  basic_histor_nthol <- tbl %>% select(nthol, nthol_return) %>%
    slice((1+i-1):(k+i-1)) %>%
    mutate(Periods_1 = 1:k) %>%
    arrange(nthol_return) %>%
    mutate(eq_weights = 1/k,
           cum_eq_weigths = cumsum(eq_weights))

  awh_nthol <- basic_histor_nthol %>%
    mutate(hybrid_weights = ((1-lambda)*(lambda^(Periods_1-1)))/(1-lambda^k),
           hybrid_cum_weights = cumsum(hybrid_weights))

  nthol_d_backtest[i,5] <- approx(x =awh_nthol$hybrid_cum_weights,
                                  y = awh_nthol$nthol_return,
                                  xout=alpha_)$y

  nthol_d[i,4] <- ifelse(approx(x =awh_nthol$hybrid_cum_weights,
                                y = awh_nthol$nthol_return,
                                xout=alpha_)$y>
                          tbl %>% select(banvt_return) %>%
                          filter(row_number() == k+i) %>% pull(),
                        0,1)
}
```

Banvit | Average Performance

VaR performances calculated by converting the dataset to daily base and calculated with 90% confidence are as follows. Parametric var and Age Weighted HS methods have been observed to give more meaningful accurate results.

```
banvt_d %>% colMeans(na.rm = T) %>%
  kable()
```

	x
parametric_VaR	0.9347826
historic_VaR	0.8695652
monte_carlo_VaR	0.8478261
awh_VaR	0.9130435

Backtests

Kupiec Tests (POF)

$H_0 : p = \hat{p}$

$H_A : p \neq \hat{p}$

Christoffesen Test

H_0 : independent failures

H_A : dependent failures

When the data is converted to daily base, it is observed that the results are more consistent than backtest methods. In addition, it is observed that the Age Weighted HS method is more preferable.

```
bind_rows(
  bind_cols(` ` = "Kupiec_banvt",
            tibble(Parametric_VaR = (BacktestVaR(banvt_d_backtest$Actual,
                                                 banvt_d_backtest$param_VaR, 0.1)$LRuc)[2],
                   Historical_VaR = (BacktestVaR(banvt_d_backtest$Actual,
                                                 banvt_d_backtest$hist_VaR, 0.1)$LRuc)[2],
                   MonteCarlo_VaR = (BacktestVaR(banvt_d_backtest$Actual,
                                                 banvt_d_backtest$mc_VaR, 0.1)$LRuc)[2],
                   AgeWeighted_VaR = (BacktestVaR(banvt_d_backtest$Actual,
                                                  banvt_d_backtest$ewhs_VaR, 0.1)$LRuc)[2])),
  bind_cols(` ` = "Christoffesen_banvt",
            tibble(Parametric_VaR = (BacktestVaR(banvt_d_backtest$Actual,
                                                 banvt_d_backtest$param_VaR, 0.1)$LRcc)[2],
                   Historical_VaR = (BacktestVaR(banvt_d_backtest$Actual,
                                                 banvt_d_backtest$hist_VaR, 0.1)$LRcc)[2],
                   MonteCarlo_VaR = (BacktestVaR(banvt_d_backtest$Actual,
                                                 banvt_d_backtest$mc_VaR, 0.1)$LRcc)[2],
                   AgeWeighted_VaR = (BacktestVaR(banvt_d_backtest$Actual,
                                                  banvt_d_backtest$ewhs_VaR, 0.1)$LRcc)[2]))) %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_banvt	0.4039993	0.5089021	0.2693894	0.7634685
Christoffesen_banvt	0.2417497	0.3094733	0.3410004	0.5701962

Net Holding | Average Performance

It was observed that the Age Weighted method exhibited over 95% success on nthol data on a daily basis.

```
nthol_d %>% colMeans(na.rm = T) %>%
  kable()
```

	x
parametric_VaR	0.9130435
historic_VaR	0.9130435
monte_carlo_VaR	0.8478261
awh_VaR	0.9565217

Backtests

According to the calculated backtest values, it can be said that the results are more meaningful and consistent than other time periods.

```
bind_rows(
  bind_cols(` ` = "Kupiec_Nthol",
            tibble(Parametric_VaR = (BacktestVaR(nthol_d_backtest$Actual,
                                                 nthol_d_backtest$param_VaR, 0.1)$LRuc)[2],
                   Historical_VaR = (BacktestVaR(nthol_d_backtest$Actual,
                                                 nthol_d_backtest$hist_VaR, 0.1)$LRuc)[2],
                   MonteCarlo_VaR = (BacktestVaR(nthol_d_backtest$Actual,
                                                 nthol_d_backtest$mc_VaR, 0.1)$LRuc)[2],
                   AgeWeighted_VaR = (BacktestVaR(nthol_d_backtest$Actual,
                                                 nthol_d_backtest$ewhs_VaR, 0.1)$LRuc)[2])),
  bind_cols(` ` = "Christoffesen_Ntna.rm = Thol",
            tibble(Parametric_VaR = (BacktestVaR(nthol_d_backtest$Actual,
                                                 nthol_d_backtest$param_VaR, 0.1)$LRcc)[2],
                   Historical_VaR = (BacktestVaR(nthol_d_backtest$Actual,
                                                 nthol_d_backtest$hist_VaR, 0.1)$LRcc)[2],
                   MonteCarlo_VaR = (BacktestVaR(nthol_d_backtest$Actual,
                                                 nthol_d_backtest$mc_VaR, 0.1)$LRcc)[2],
                   AgeWeighted_VaR = (BacktestVaR(nthol_d_backtest$Actual,
                                                 nthol_d_backtest$ewhs_VaR, 0.1)$LRcc[2])))) %>%
  kable()
```

	Parametric_VaR	Historical_VaR	MonteCarlo_VaR	AgeWeighted_VaR
Kupiec_Nthol	0.7634685	0.7634685	0.2693894	0.0348557
Christoffesen_Ntna.rm = Thol	0.5701962	0.6465243	0.0826700	0.1055210

Conclusion

VaR values were calculated on different data sets at different significance levels and their accuracy was checked.

- It is not possible to make a clear comment about the accuracy of VaR values obtained with frequencies higher than 30 minutes in the data set.
- At the level of 95% confidence, monte carlo method has given more accurate and reliable results than others.
- It was seen that the historical VaR value produced more accurate results when the confidence level was increased and the risk was reduced.
- When the data was converted to daily base, Age Weighted HS method, which did not stand out in the analysis calculated with other frequencies, was found to be more successful than other methods.

As a result, calculating VaR values on a daily basis will provide more accurate results. In further studies, it will be possible to find hyperparametry of Age-Weighted HS method and obtain more accurate results with correct **lambda** value.