Financial Econometrics, Homework II *

Zhihan Zhang, Jingwan LUO *Toulouse School of Economics*

This document firstly simulate a Heston Model, then study the finte and infinite sample properties; use data from 'TrueFX' Eur-US exchange data by seconds, we explore integrated volatility at different frequencies.

Keywords: Heston Model, High Frequency Data, Volatility, Simulations

Question Statement

- Consider the following simulation design:
  - The log prices $X_t = \log(S_t)$ has a true distribution which follows Itô process $dX_t = \mu dt + \sigma_t dW_t$, with $\mu = 0.01$;
  - Microstructure noises are addictive: $Y_{t_i} = X_{t_i} + \epsilon_{t_i}$, with $\epsilon_t \sim \mathcal{N}(0, V_\epsilon)$;
  - Consider different levels of noises where $\xi^2 = \frac{V_\epsilon}{\sqrt{T\int_0^T \sigma_s^2 ds}}$:
    - $\xi^2 = 0.001$
    - $\xi^2 = 0.01$
    - $\xi^2 = 0.1$ Volatility follows Heston (1993):

$$d\sigma_t^2 = \kappa\left(\alpha - \sigma_t^2\right)dt + \gamma\sigma_t dW_t$$

- where:
  - σ_t^2: volatility
  - γ: volatility of volatility; in the question context $\gamma = \frac{0.5}{252}$.
  - α: long-term price variance; $\alpha = \frac{0.04}{252}$.
  - κ: rate of reversion of the long-term price variance; $\kappa = \frac{5}{252}$.

Goal: Study the finite sample properties and the asymptotic properties of $\int_0^T \sigma_s^2 ds$.

Discretization before coding:

Based on this websource and lectures, the discretizations before implementation:

$$X_{t+\Delta} - X_t = \mu\Delta + \sigma_t(W_{t+\Delta} - W_t) \quad \sigma_{t+\Delta}^2 - \sigma_t^2 = \kappa(\alpha - \sigma_t^2)\Delta + \gamma\sigma_t(W_{t+\Delta} - W_t)$$

Since $W_{t+\Delta}$ is Wiener process, then by definition of it, we know $W_{t+\Delta} - W_t \sim \mathcal{N}(0, \Delta)$:

$$X_{t+\Delta} = X_t + \mu\Delta + \sigma_t \cdot \mathcal{N}(0,\Delta) \quad \sigma_{t+\Delta}^2 = \sigma_t^2 + \kappa(\alpha - \sigma_t^2)\Delta + \gamma\sigma_t \cdot \mathcal{N}(0,\Delta)$$

Simulation

Pre-Simulation Preparation

Typically, a trading day has 6.5 hours of opening hours (from 9:30 a.m. to 4 p.m.). Our study period t is then the duration of 23,400 seconds, which gives a time delta of $\frac{1}{t}$. We first try a number of simulations of 100. Risk-free rate, which is the theoretical rate of return of an asset with zero risk, is exogenously given. According to **?** and slides,

$$dX_t = \mu dt + \sigma_t d\mathcal{W}_{x,t}^{\mathbb{P}} \quad d\sigma_t^2 = \kappa(\alpha - \sigma_t^2)dt + \gamma\sigma_t d\mathcal{W}_{\sigma,t}^{\mathbb{P}}$$

*All project file at (link here). All authors from Toulouse School of Economics, Master 2 Econometrics and Empirical Economics Track, **Author 1**: Zhihan Zhang (ID: 21613596), **Author 2**: Jingwan Luo (ID: 22007276)

Then the bivariate stochastic processes have the following relationship:

$$\mathbb{E}[\mathcal{W}^{\mathbb{P}}_{x,t}, \mathcal{W}^{\mathbb{P}}_{\sigma,t}] = \rho dt \quad \mathcal{W}^{\mathbb{P}}_{\sigma,t} = \rho \mathcal{W}^{\mathbb{P}}_{x,t} + \sqrt{1-\rho^2} B_t$$

We chose $\rho = 0.7$. According to the discretisation above, `sdel` stands for $\sqrt{\Delta}$.

```
# Later, we change sample size here
t = 23400                          # number of seconds in a 6.5 hour trading day
n = 100                                                  # number of simulations
del = 1/t                                                              # time delta

mu = 0.01                                                          # risk-free rate

kappa = 5/252                                                    # rate of reversion
alpha = 0.04/252                                        # long-term price variance
gamma = 0.5/252                                         # volatility of volatility

rho = 0.7                          # correlation between the two Wiener processes

sdel = sqrt(del)                                       # N(0, sdel^2), discretisation
srho = sqrt(1-rho^2)       # under W_{2,t} = \rho W_{1,t} + \sqrt{(1-\rho^2)}B_t
```

To satisfy Feller's condition to make the zero boundary unattainable by the volatility process:

$$2\kappa\alpha \geq \gamma^2$$

```
# check feller's condition:

if(2*kappa*alpha >= gamma^2){
  print("Feller's Condition is satisfied with the given parameter.")
} else{
  print("Feller's Condition is not satisfied with the given parameter")
}
```

```
## [1] "Feller's Condition is satisfied with the given parameter."
```

We prepared the matrices for asset pricing and volatility:

```
S0 = 100                                                    # initial asset price
X0 = log(S0)                                          # log of initial asset price
v0 = 10^(-3)                                                 # initial volatility

blanks=matrix(0,                # matrix for setting initial values of X0 and v0
              nrow=n,
              ncol=(t-1))
x = cbind(matrix(X0,                                  # first col filled in with X0
                 nrow=n,
                 ncol=1), blanks)
s = cbind(matrix(v0,                                  # first col filled in with v0
                 nrow=n,
                 ncol=1), blanks)
```

and generate the two bivariate Wiener processes $\mathcal{W}^{\mathbb{P}}_{x,t}$ and $\mathcal{W}^{\mathbb{P}}_{s,t}$ accroding to the choice of ρ:

```
W1 = matrix(rnorm(n*t,
                  mean = 0,
                  sd = sdel),         # Wiener process with asset price, W_{1,t}
            nrow=n,
            ncol=t)
Bt = matrix(rnorm(n*t,
                  mean = 0,
                  sd = sdel),
            nrow=n,
            ncol=t)
W2 = (Bt*srho +rho*W1)                 # Wiener process with volatility, W_{2,t}
```

Simulations

simulations for X_t and σ_t^2 according to the stochastic differential equations:

```
for (i in 1:(t-1)){
  s[,i+1] <- s[,i] +kappa*(alpha - s[,i])*del + gamma*sqrt(s[,i])*W2[,i]
  x[,i+1] <- x[,i] + (mu)*del + sqrt(s[,i])*W1[,i]
}
```

Microstructures

In discretisation, the noise ϵ_t has a variance of V_ϵ:

$$V_\epsilon = \xi^2 \cdot \sqrt{\int_0^T \sigma_s^2 ds} = \xi^2 \cdot \sqrt{\sum_s \sigma_s^2 \cdot \Delta}$$

```
xi1 = 0.001
v_e1 = xi1^2*sqrt(sum(s)*del)
noise1 = rnorm(n*t,
               mean = 0,
               sd = sqrt(v_e1))
y1 = x + noise1
```

```
xi2 = 0.01
v_e2 = xi2*sqrt(sum(s)*del)
noise2 = rnorm(n*t,
               mean = 0,
               sd = sqrt(v_e2))
y2 = x + noise2
```

```
xi3 = 0.1
v_e3 = xi3*sqrt(sum(s)*del)
noise3 = rnorm(n*t,
               mean = 0,
               sd = sqrt(v_e3))
y3 = x + noise3
```

Estimation of integrated volatility

The standard estimator for $\int_0^t \sigma_s^2 ds$ is:

$$RV = \sum_{t_{i+1} \leq t} (X_{t_{i+1}} - X_{t_i})^2$$

We then construct the estimator RV for X_t:

```
RV = sum(diff(x)^2)
RV
```

```
## [1] 1836.453
```

With noisy data $Y_i = X_i + \epsilon_i$, the expected value of RV:

$$\mathbb{E}[RV] = \int_0^t \sigma_s^2 ds + 2n\mathbb{V}ar(\epsilon)$$

The variance of noise for different level of ξ:

```
Varepsilon1 = var(noise1)
Varepsilon2 = var(noise2)
Varepsilon3 = var(noise3)
expectedRV1 = sum(s) + 2*t*var(noise1)
expectedRV2 = sum(s) + 2*t*var(noise2)
expectedRV3 = sum(s) + 2*t*var(noise3)
expectedRV1
```

```
## [1] 2328.474
```

```
expectedRV2
```

```
## [1] 2476.138
```

```
expectedRV3
```

```
## [1] 3803.153
```

- Observation:
  - the standard estimator for RV is biased;
  - as ξ grows larger, the bias is larger;

Now we change the sample size t to study its asymptotic behavior. With $10t$ of a sample size, the estimator remain inconsistent.

Exercise:

Use the exchange rates high-frequency data from "TrueFX" to compute 04 different daily estimators of the integrated volatility at different frequencies: 1s, 10s, 1minute, 5 minutes, 10 minutes.

Necessary libraries loaded:

```
library(quantmod)
```

```
## Warning: package 'zoo' was built under R version 4.1.2
```

```
library(TTR)
library(PerformanceAnalytics)
library(tseries)
library(zoo)
library(xts)
library(dplyr)
library(knitr)
library(highfrequency)
```

```
## Warning: package 'highfrequency' was built under R version 4.1.2
```

```
library(rtsdata)
library(lubridate)
```

Use European and U.S. exchange data by seconds on month December of 2021.

```
eurus <- read.csv('~/Desktop/Financial Econometrics/EURUSD-2021-12.csv',
                  header=FALSE)
datetime <- strptime(eurus[,2], format="%Y%m%d %H:%M:%OS")
eurus <- as.xts(eurus[,3:4], order.by=datetime)
colnames(eurus) <- c("bid","ask")
head(eurus)
```

```
##                          bid     ask
## 2021-12-01 00:00:00 1.13298 1.13305
## 2021-12-01 00:00:00 1.13299 1.13305
## 2021-12-01 00:00:00 1.13300 1.13305
## 2021-12-01 00:00:00 1.13299 1.13305
## 2021-12-01 00:00:00 1.13298 1.13305
## 2021-12-01 00:00:00 1.13299 1.13302
```

Calculate the midrate:

```
midrate <- (eurus[,1] + eurus[,2])/2
names(midrate) <- "midpoint"
```

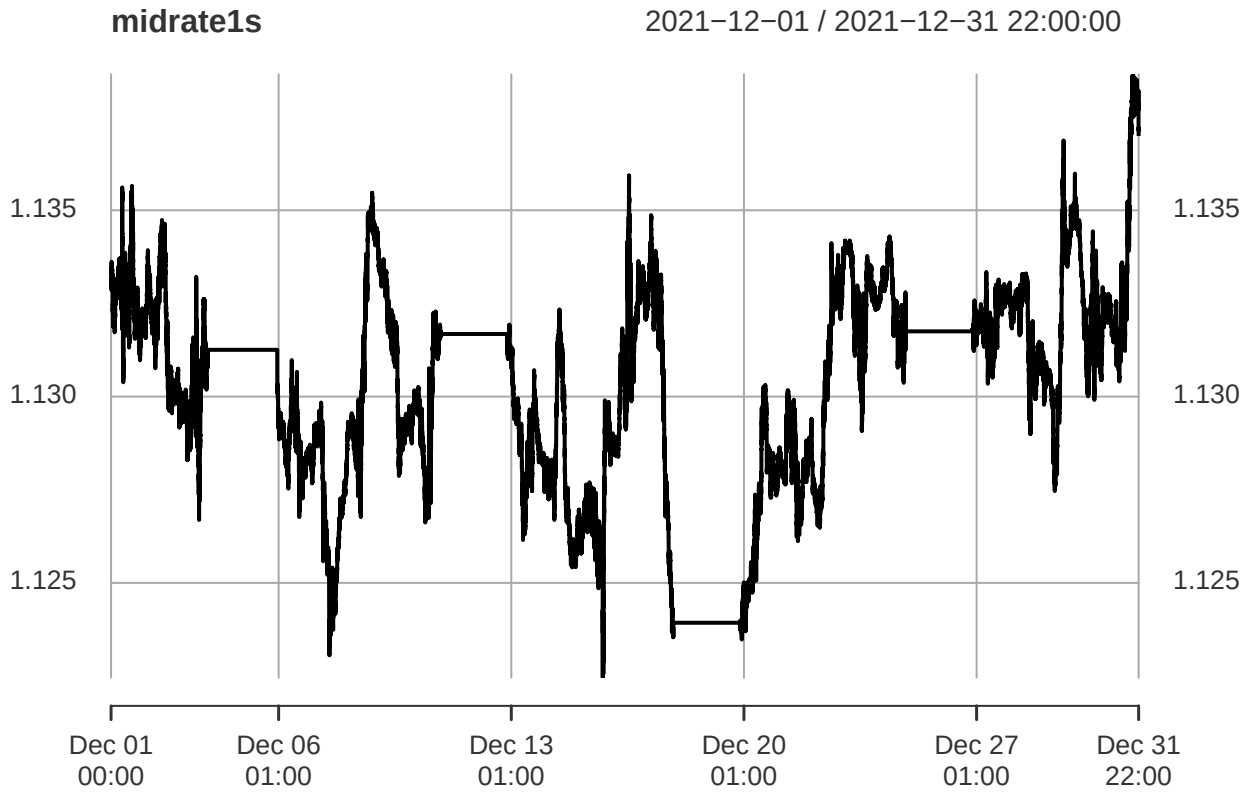
Aggregate the midrate to various frequencies.

1 second aggregation, remove weekends and calculate returns:

```
midrate1s <- aggregateTS(midrate, on="seconds", k=1)
head(midrate1s)
```

```
##                     midpoint
## 2021-12-01 00:00:00 1.133015
## 2021-12-01 00:01:00 1.132955
## 2021-12-01 00:02:00 1.132945
## 2021-12-01 00:03:00 1.132880
## 2021-12-01 00:04:00 1.133075
## 2021-12-01 00:05:00 1.132965
```

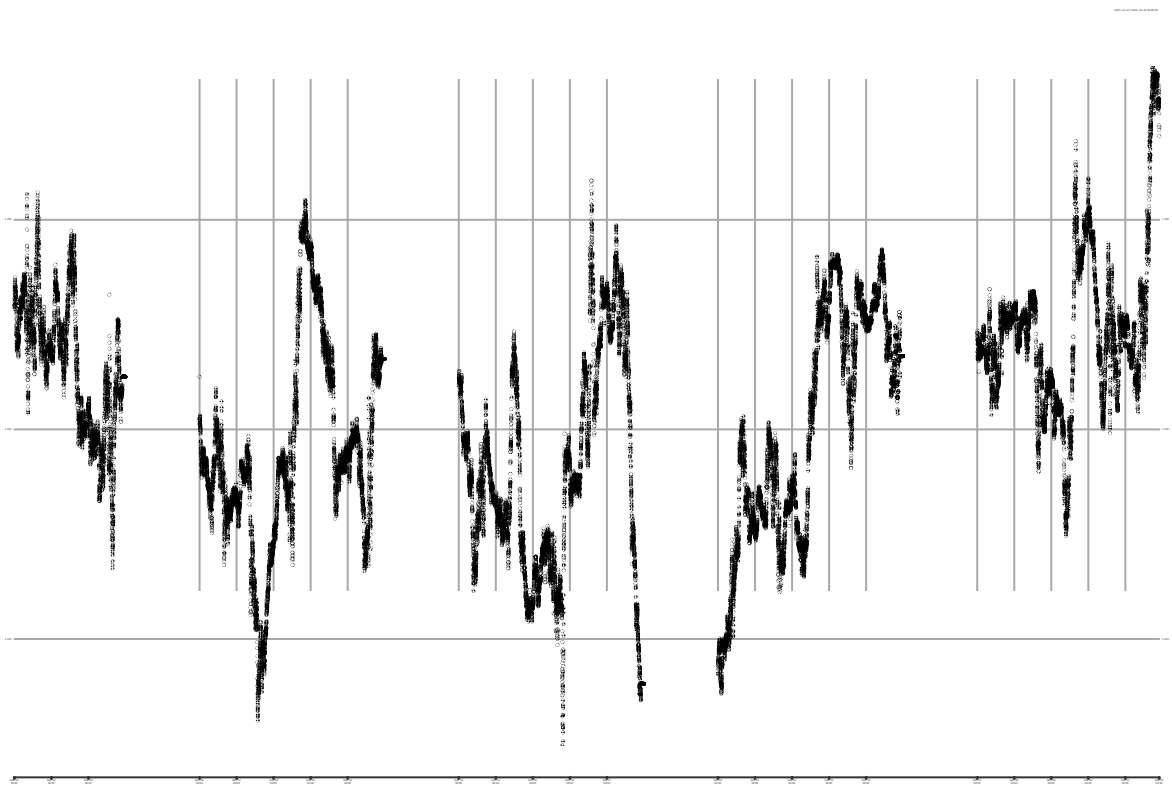
```
plot(midrate1s)
```




```
index <- .indexwday(midrate1s)
unique(index)
```

```
## [1] 3 4 5 6 0 1 2
```

```
midrate1s <- midrate1s[index %in% 1:5]
plot(midrate1s, main="", type="p", pch=1, cex=0.05)
```

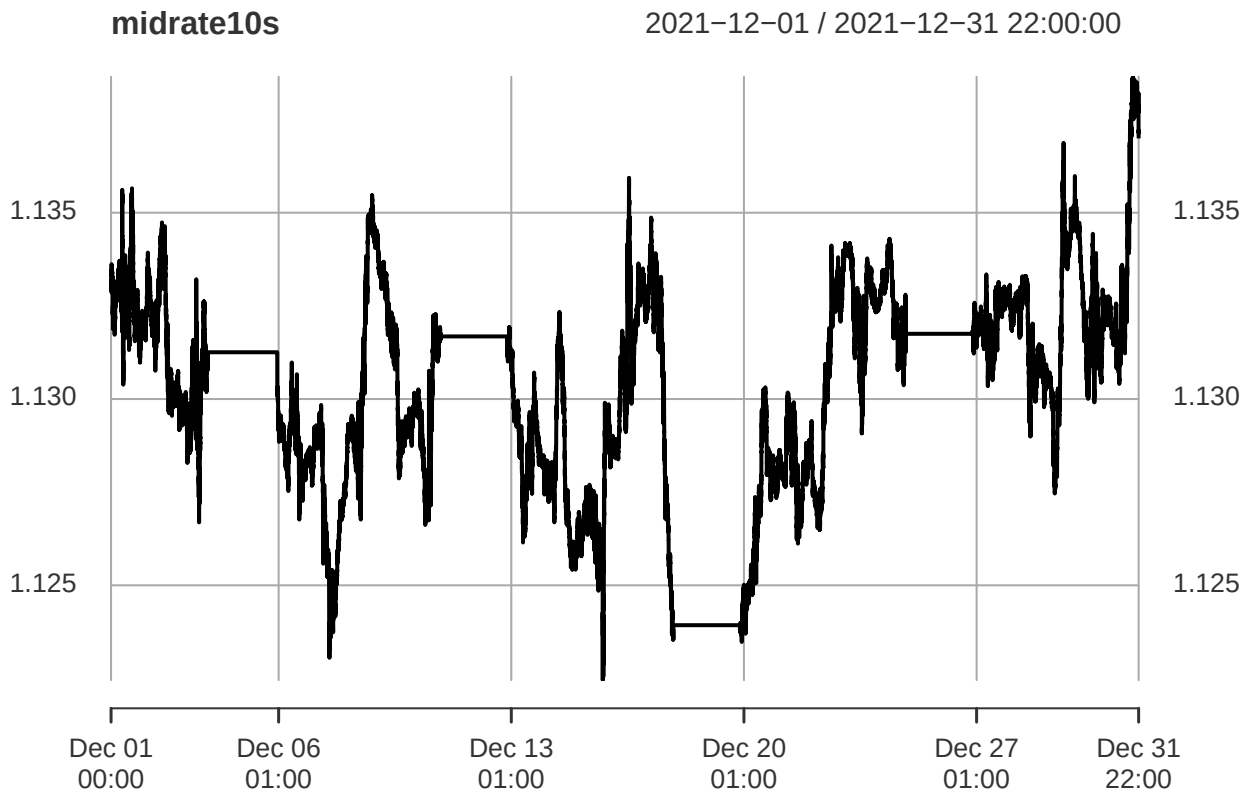


```
ret1s <- 100 * diff(log(midrate1s))
summary(ret1s)
```

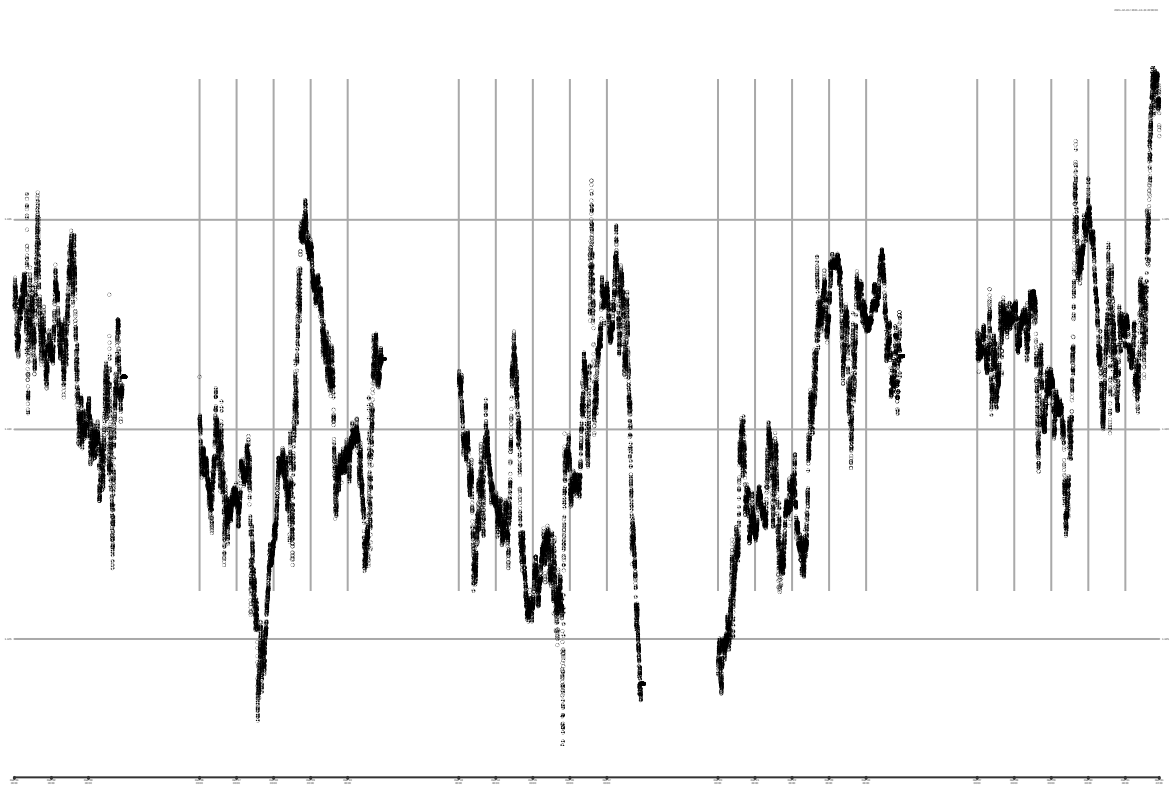
```
##       Index                          midpoint
##  Min.   :2021-12-01 00:00:00   Min.   :-0.2485434
##  1st Qu.:2021-12-08 17:30:00   1st Qu.:-0.0039854
##  Median :2021-12-16 11:00:00   Median : 0.0000000
##  Mean   :2021-12-16 19:01:54   Mean   : 0.0000106
##  3rd Qu.:2021-12-24 04:30:00   3rd Qu.: 0.0039856
##  Max.   :2021-12-31 22:00:00   Max.   : 0.3531597
##                                NA's   :1
```

Similarly, for 10 seconds aggregation:

```
midrate10s <- aggregateTS(midrate, on="seconds", k=10)
head(midrate10s)
plot(midrate10s)
```



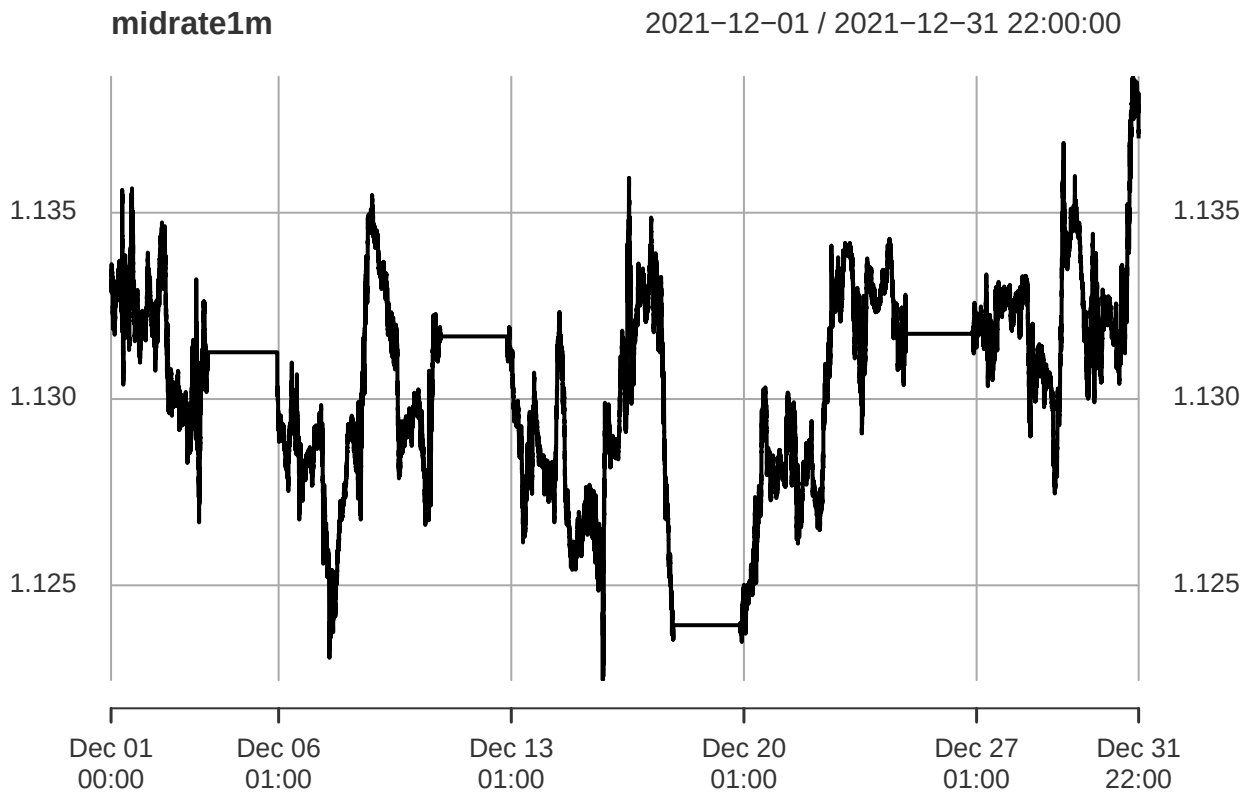

```
index <- .indexwday(midrate10s)
unique(index)
midrate10s <- midrate10s[index %in% 1:5]
plot(midrate10s, main="", type="p", pch=1, cex=0.05)
```



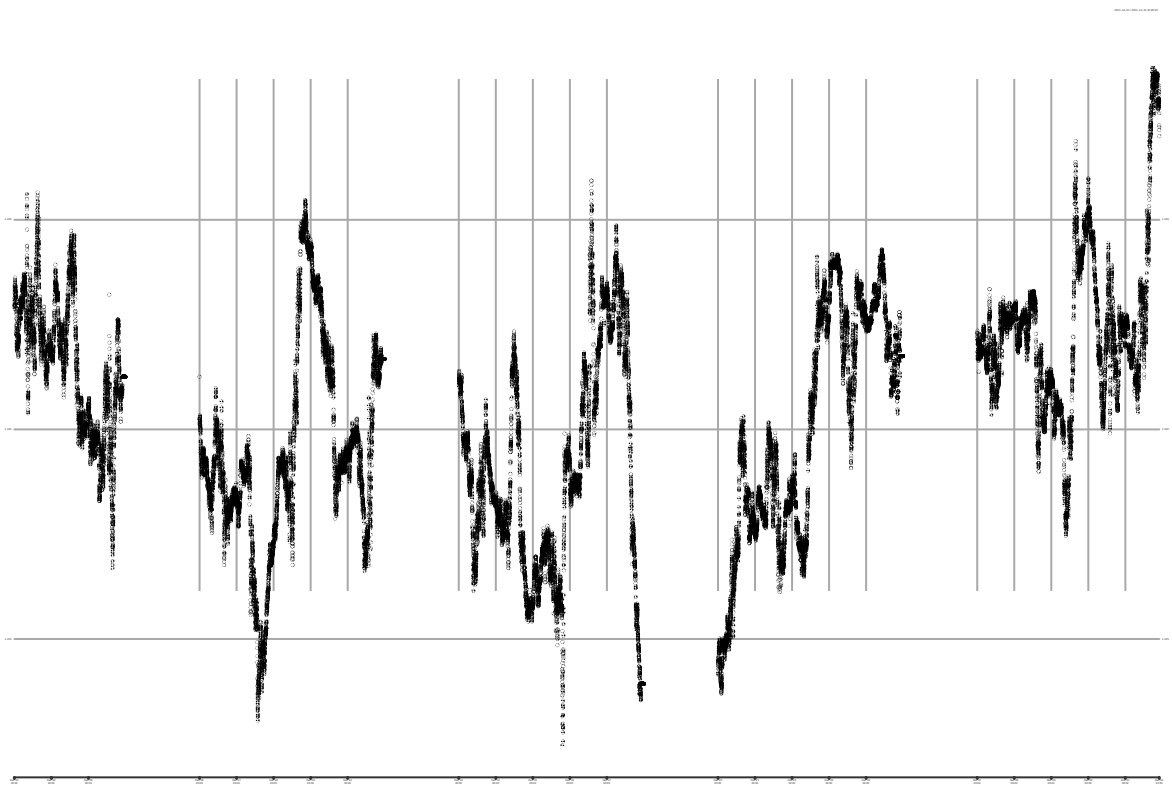
```
ret10s <- 100 * diff(log(midrate10s))
summary(ret10s)
```

For one minute aggregation:

```
midrate1m <- aggregateTS(midrate, on="minutes", k=1)
head(midrate1m)
plot(midrate1m)
```



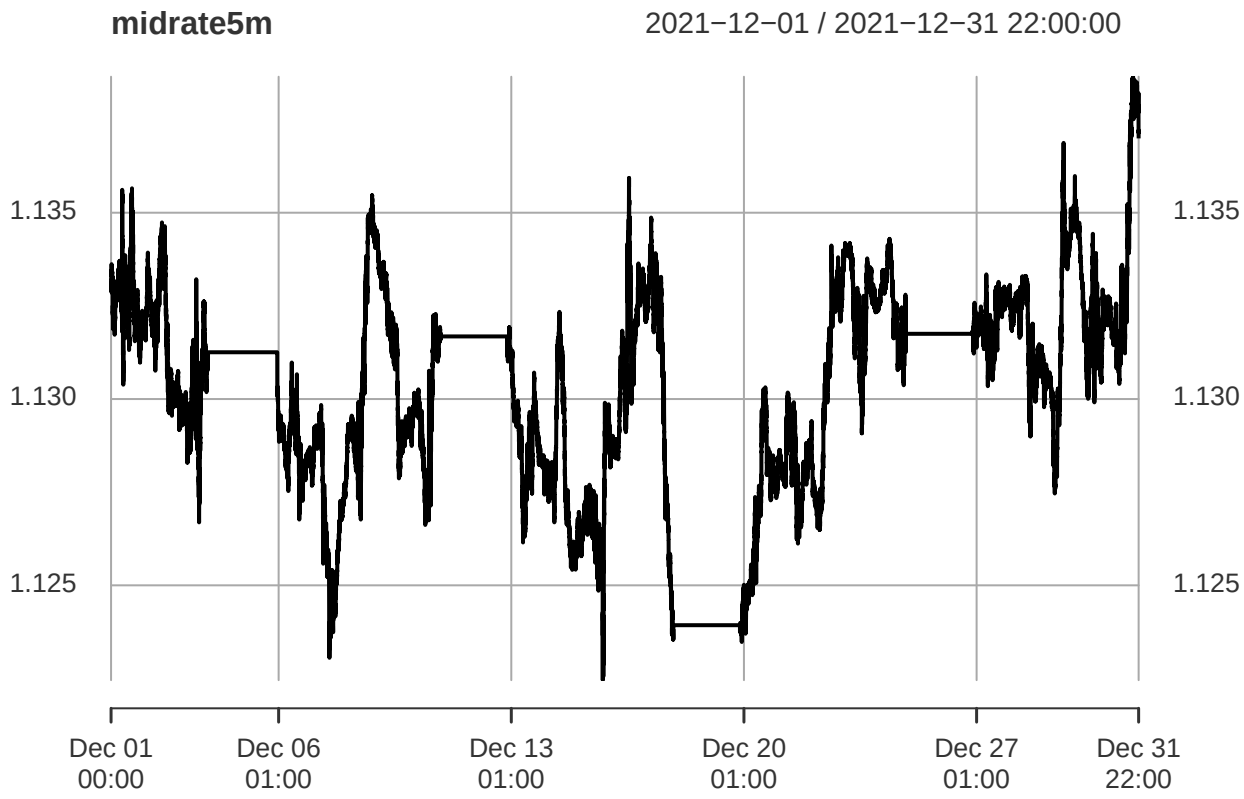

```
index2 <- .indexwday(midrate1m)
unique(index2)
midrate1m <- midrate1m[index2 %in% 1:5]
plot(midrate1m, main="", type="p", pch=1, cex=0.05)
```



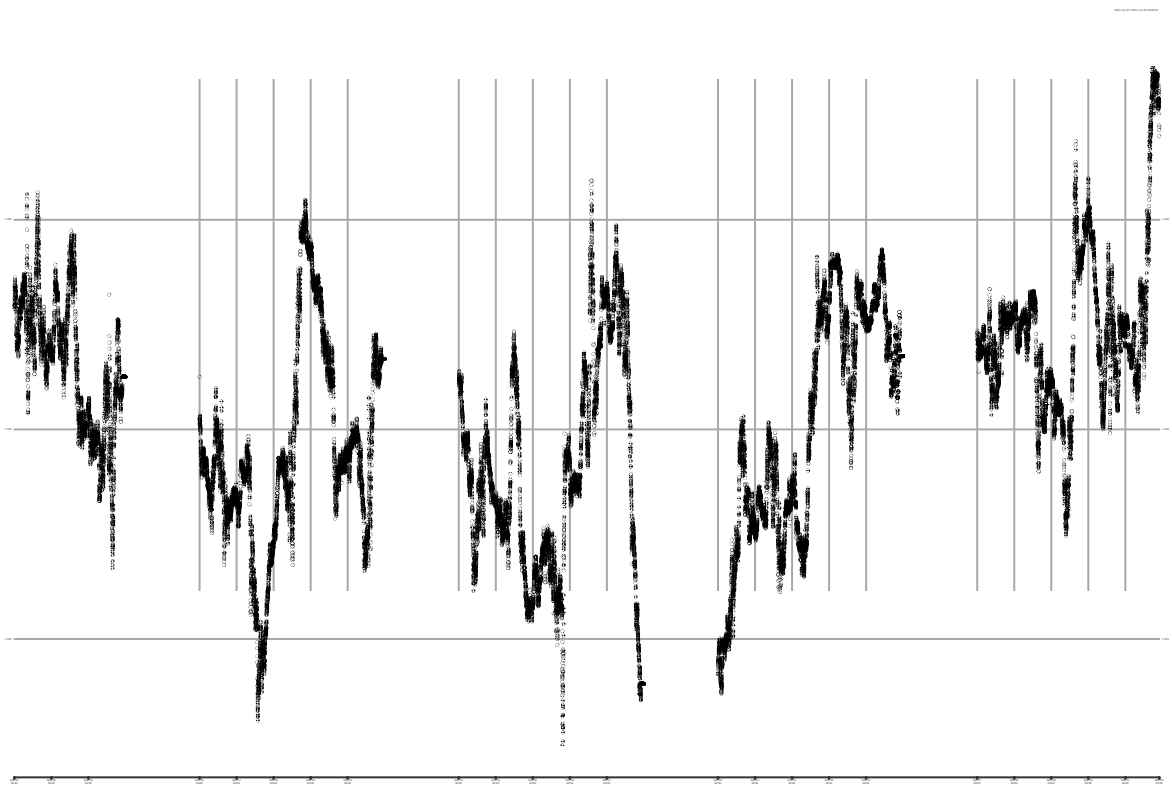
```
ret1m <- 100 * diff(log(midrate1m))
summary(ret1m)
```

For 5 minutes aggregation:

```
midrate5m <- aggregateTS(midrate, on="minutes", k=5)
head(midrate5m)
plot(midrate5m)
```



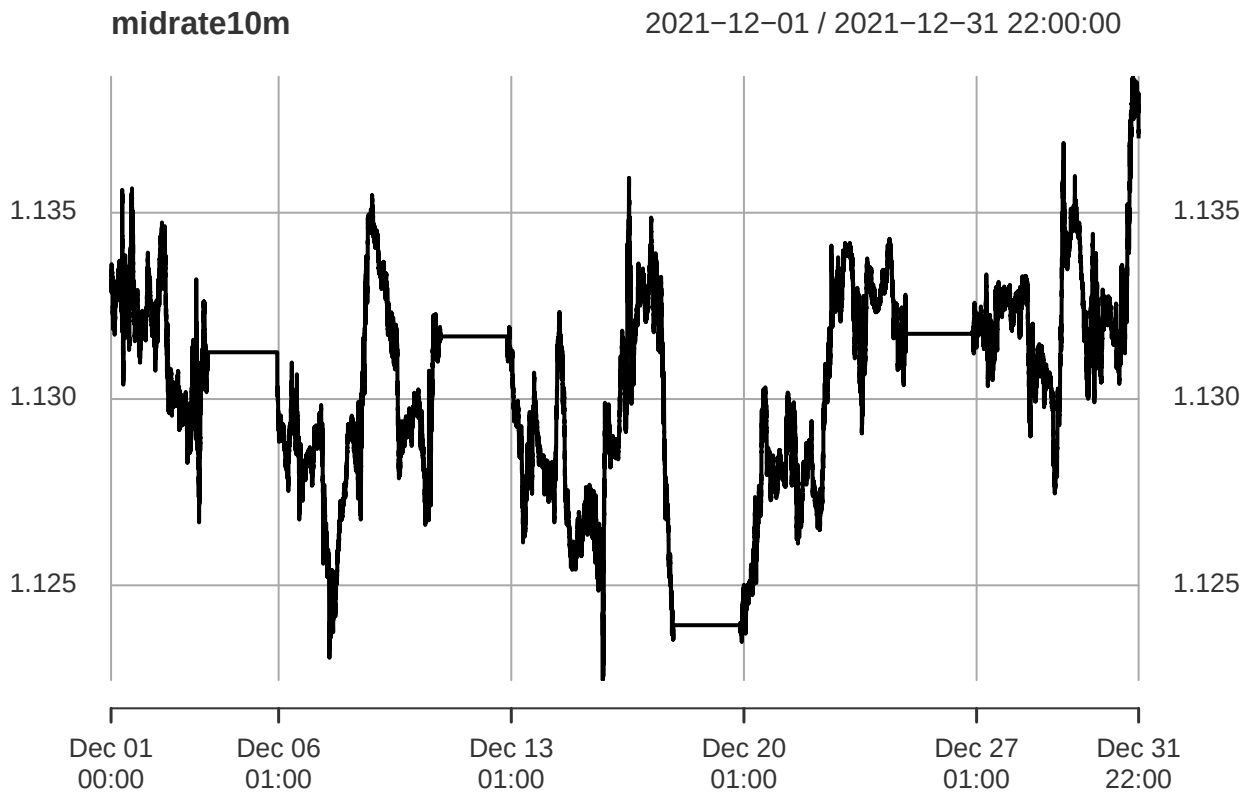

```
index2 <- .indexwday(midrate5m)
unique(index2)
midrate5m <- midrate5m[index2 %in% 1:5]
plot(midrate5m, main="", type="p", pch=1, cex=0.05)
```



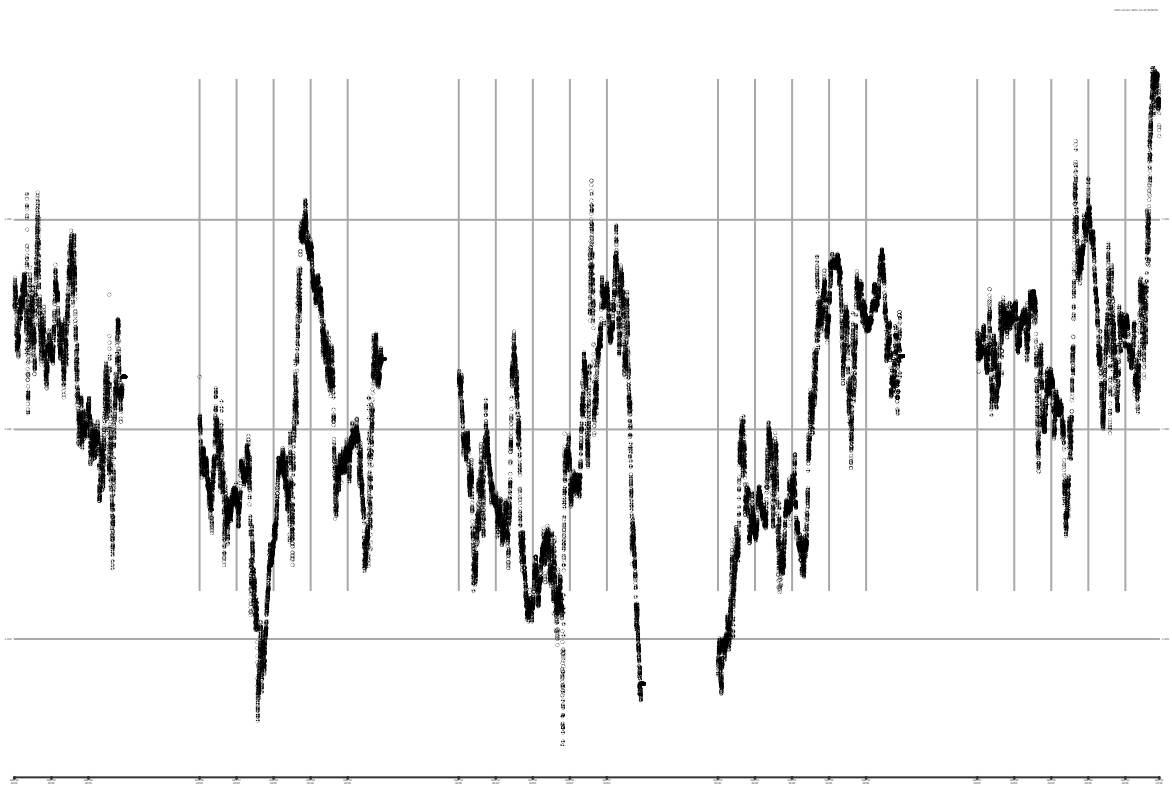
```
ret5m <- 100 * diff(log(midrate5m))
summary(ret5m)
```

For 10 minutes aggregation:

```
midrate10m <- aggregateTS(midrate, on="minutes", k=10)
head(midrate10m)
plot(midrate10m)
```




```
index2 <- .indexwday(midrate10m)
unique(index2)
midrate10m <- midrate10m[index2 %in% 1:5]
plot(midrate5m, main="", type="p", pch=1, cex=0.05)
```

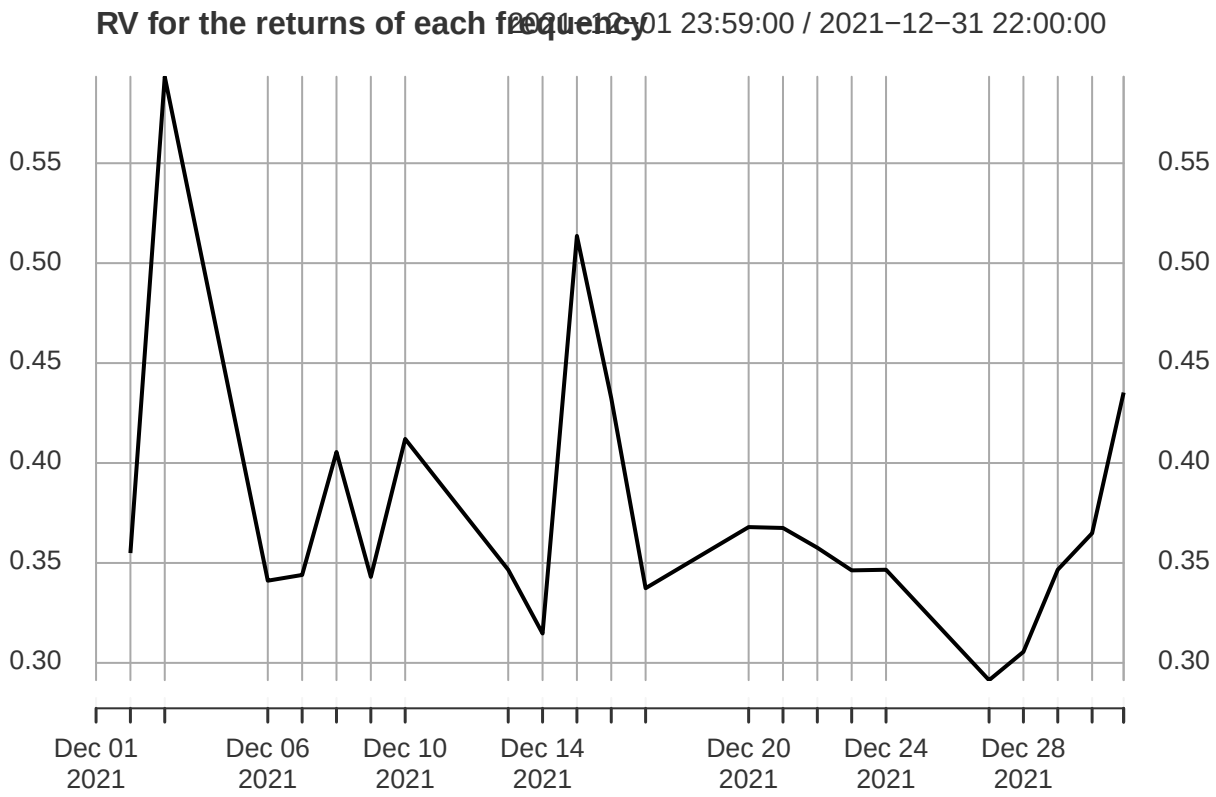


```
ret10m <- 100 * diff(log(midrate10m))
summary(ret10m)
```

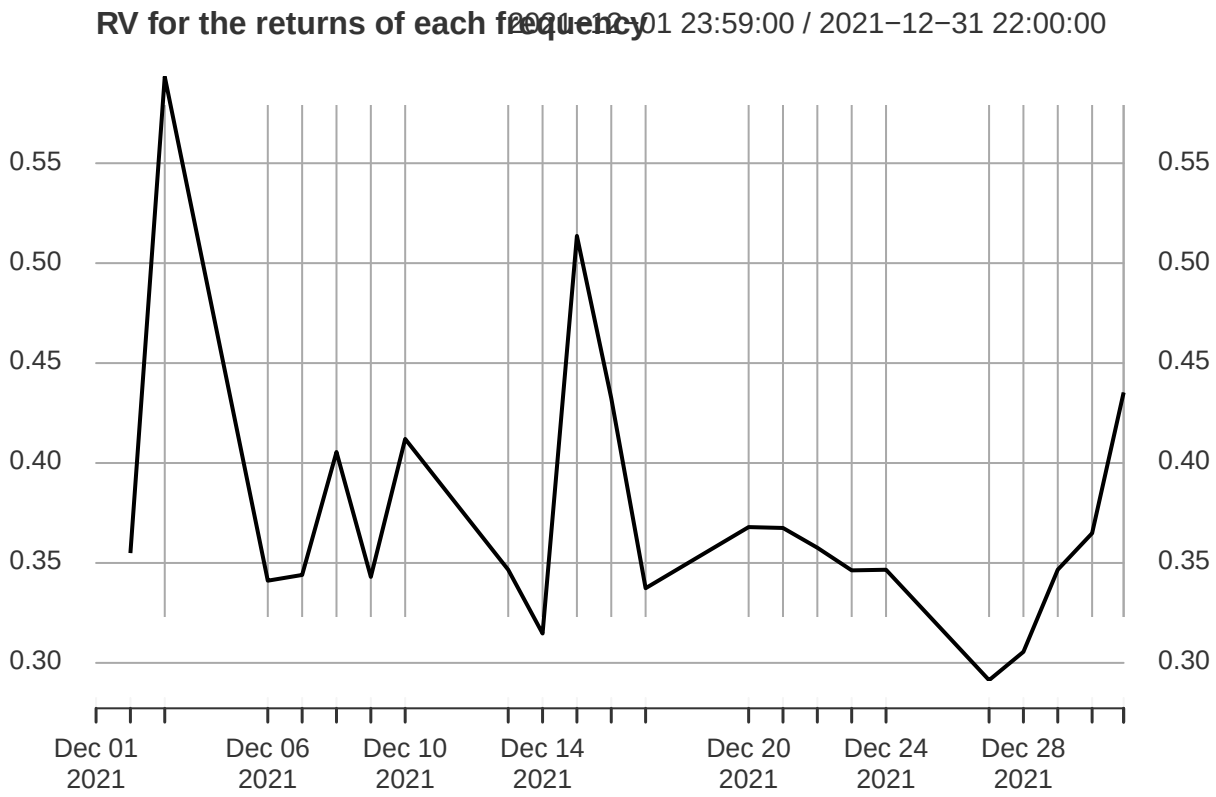
Different Estimator Calculations

Realised Covariance:

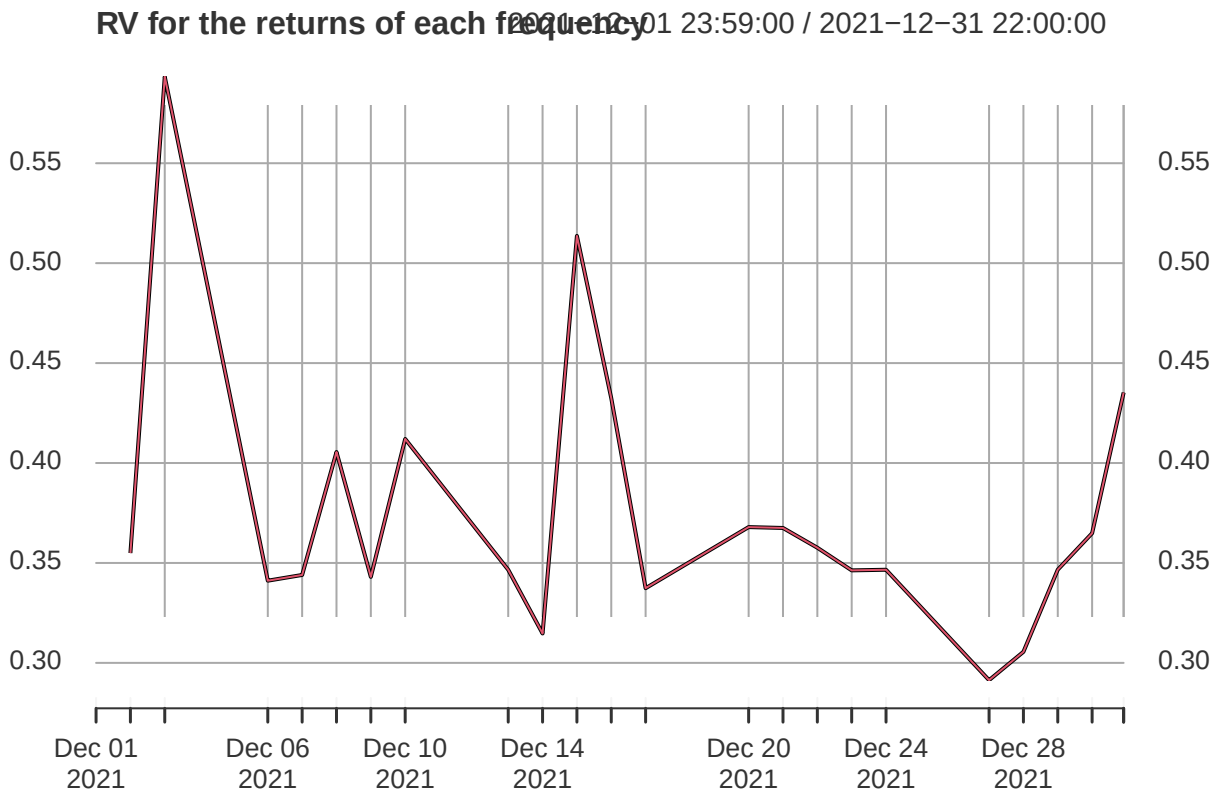
```
rv1s <- rCov(ret1s, align.by="seconds", align.period=1)
rv10s <- rCov(ret10s, align.by = "seconds", align.period=10)
rv1m   <- rCov(ret1m, align.by="minutes", align.period=1)
rv5m   <- rCov(ret5m, align.by="minutes", align.period=5)
rv10m <- rCov(ret10m, align.by="minutes", align.period=10)

plot(rv1s^0.5, main = "RV for the returns of each frequency")
```



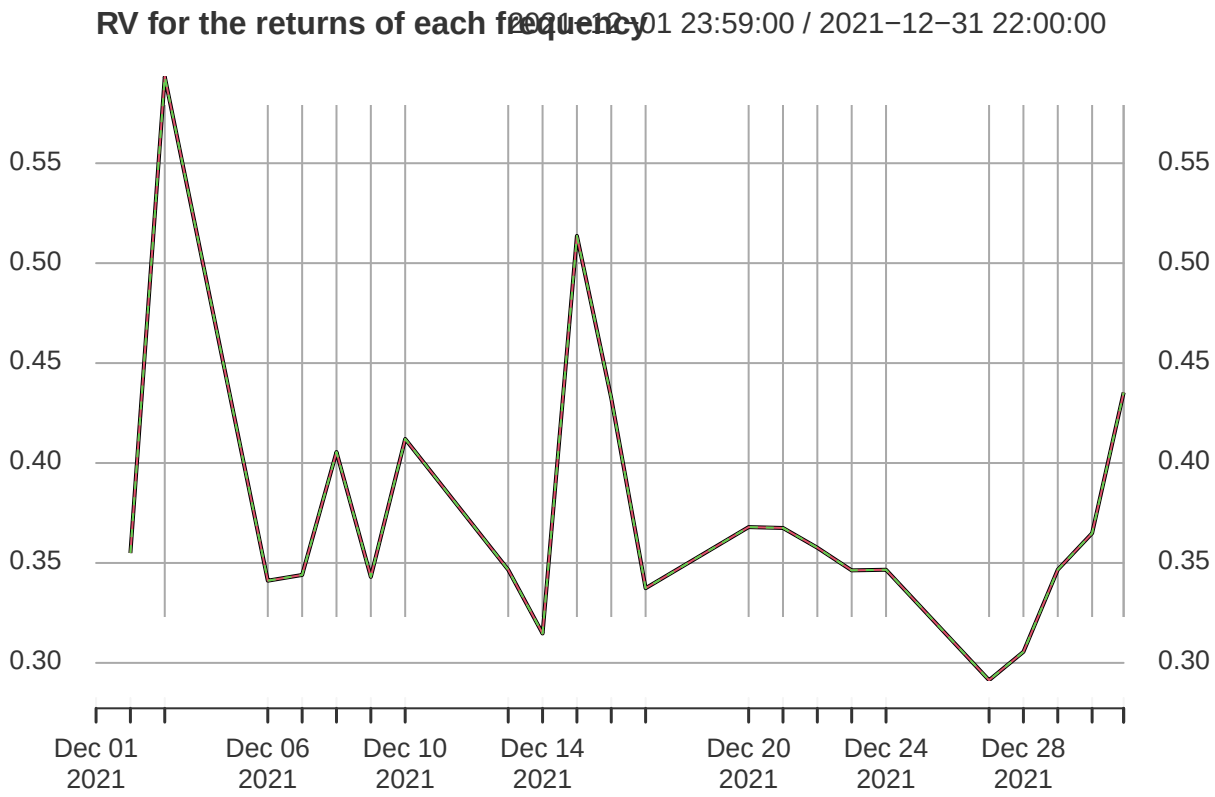

```
lines(rv10s^0.5, col = 1)
```



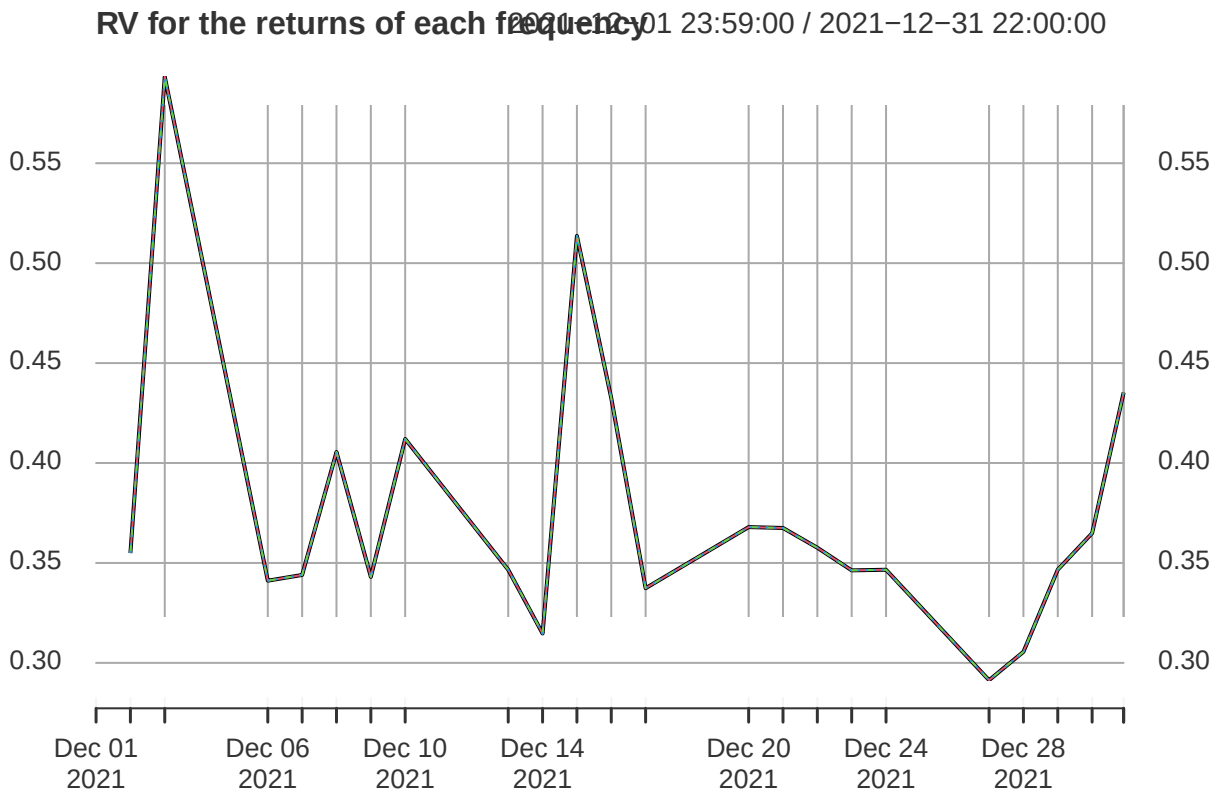

```
lines(rv1m^0.5, col = 2, lty = 1)
```




```
lines(rv5m^0.5, col = 3, lty = 2)
```

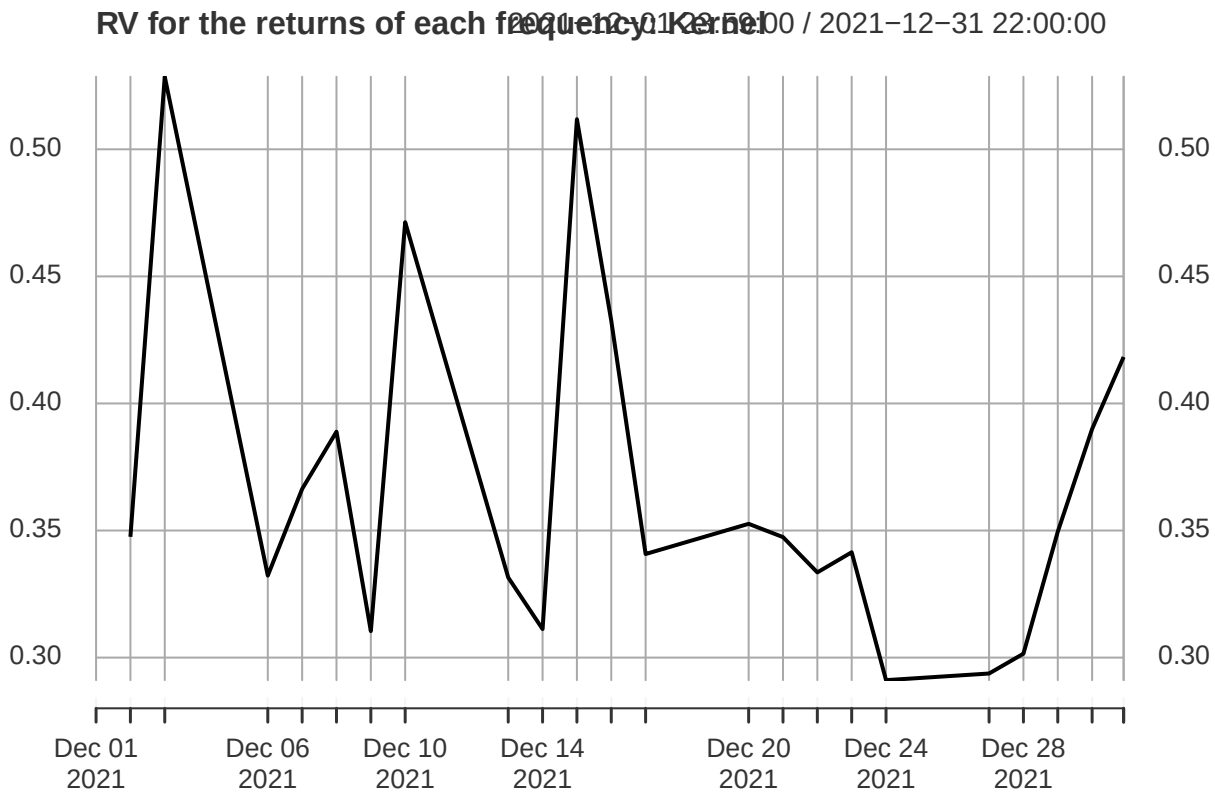



```
lines(rv10m^0.5, col = 4, lty = 3)
```

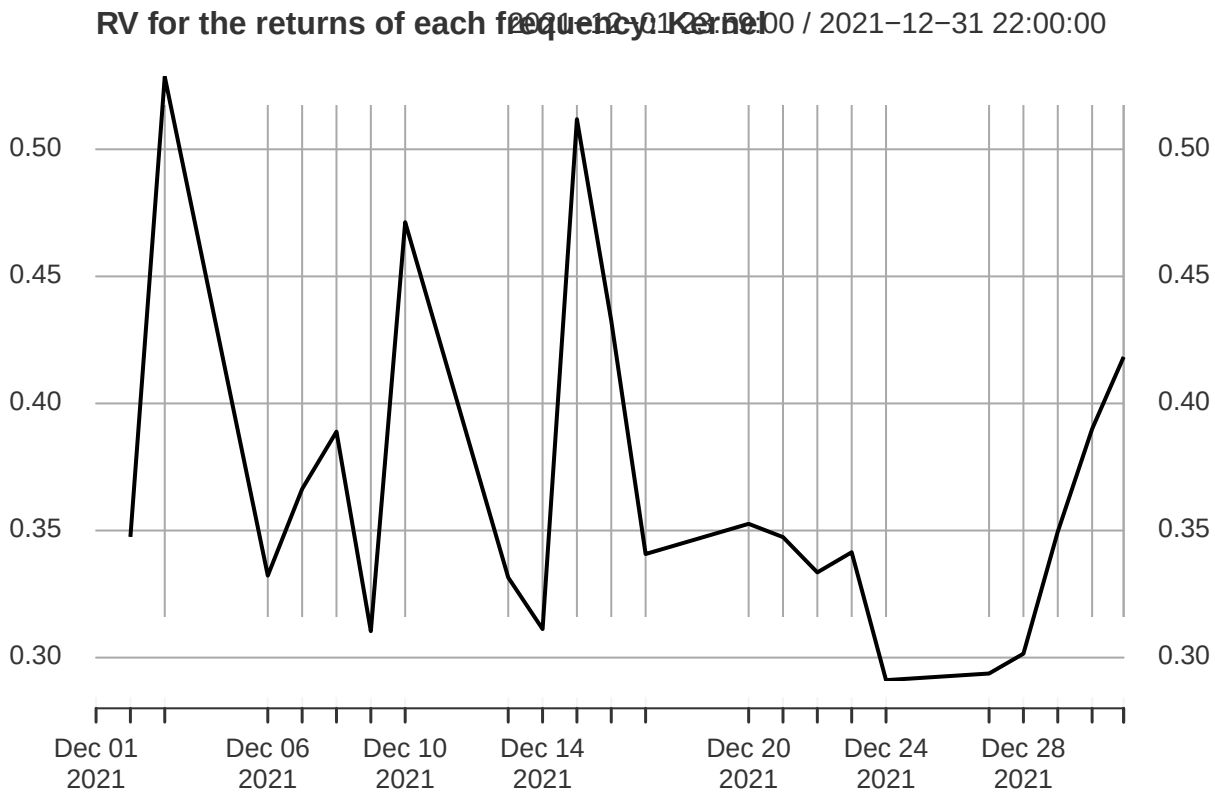


Kernel Estimator:

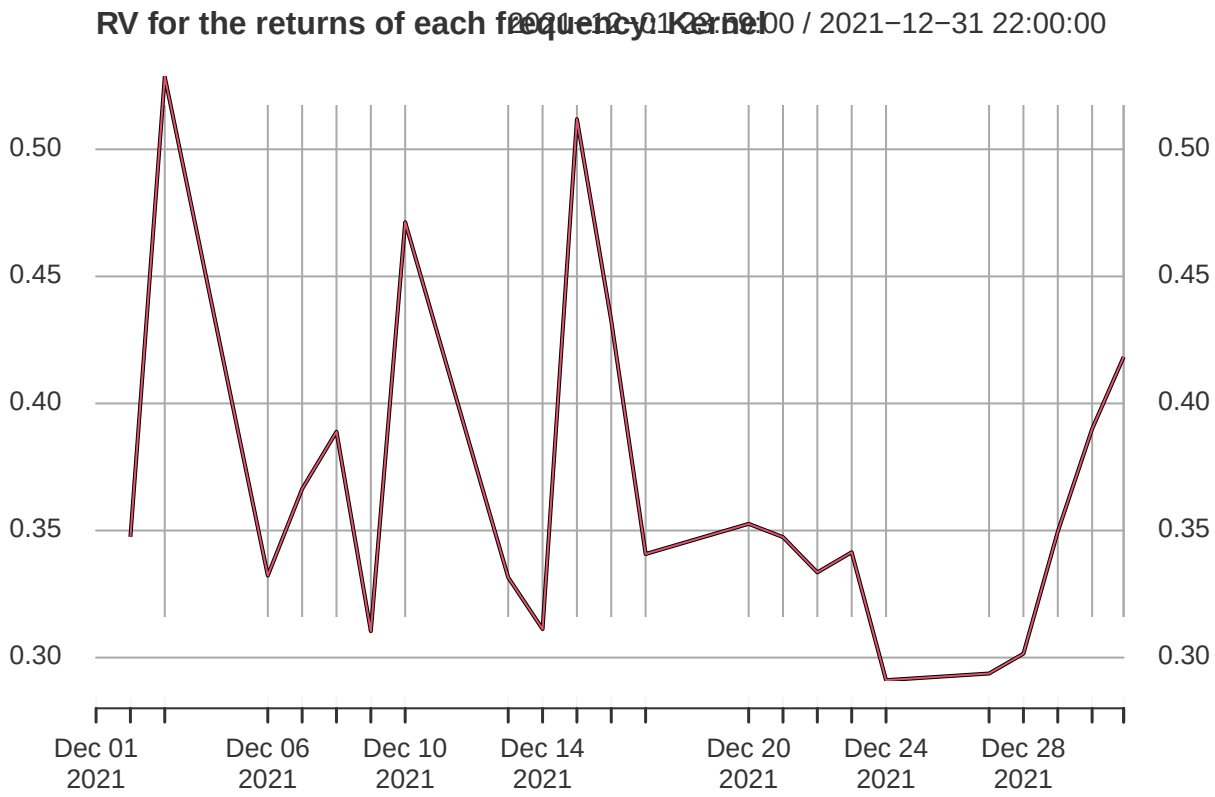
```
rv1s_K <- rKernelCov(ret1s, align.by="seconds", align.period=1)
rv10s_K <- rKernelCov(ret10s, align.by = "seconds", align.period=10)
rv1m_K   <- rKernelCov(ret1m, align.by="minutes", align.period=1)
rv5m_K   <- rKernelCov(ret5m, align.by="minutes", align.period=5)
rv10m_K <- rKernelCov(ret10m, align.by="minutes", align.period=10)

plot(rv1s_K^0.5, main = "RV for the returns of each frequency: Kernel")
```



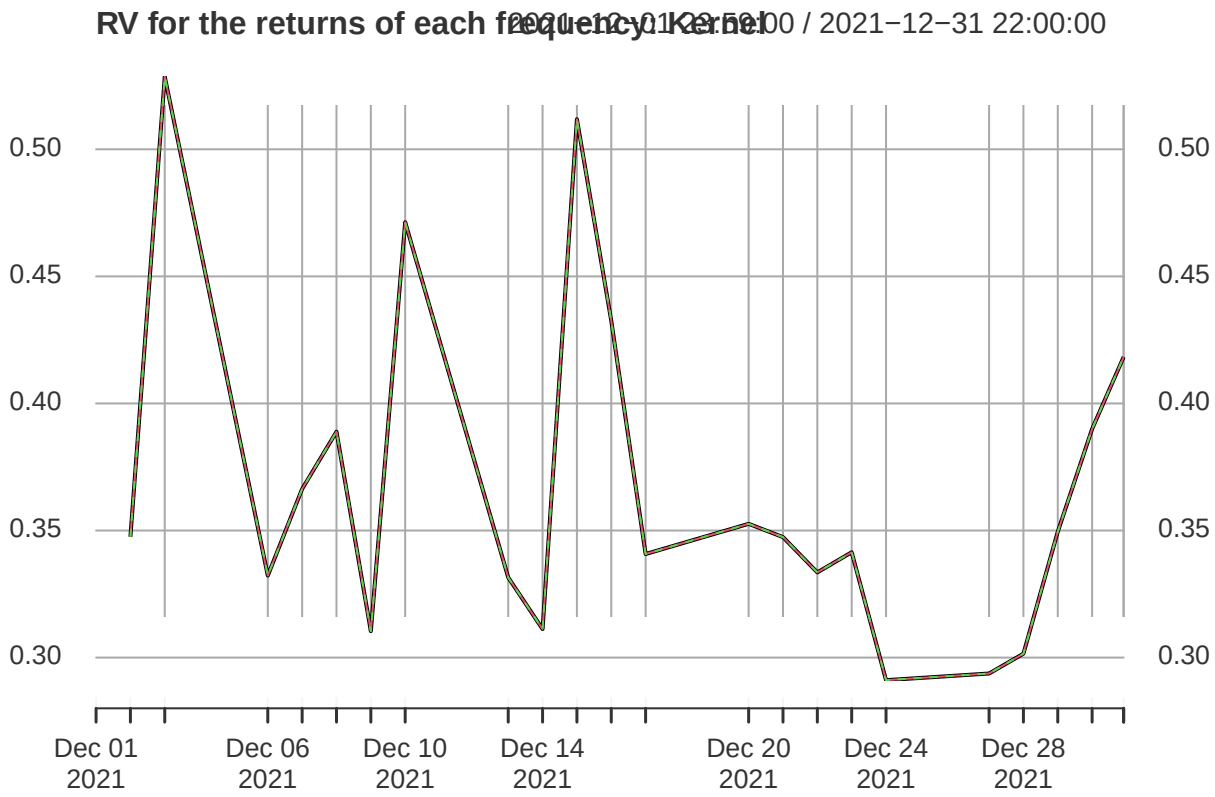

```
lines(rv10s_K^0.5, col = 1)
```



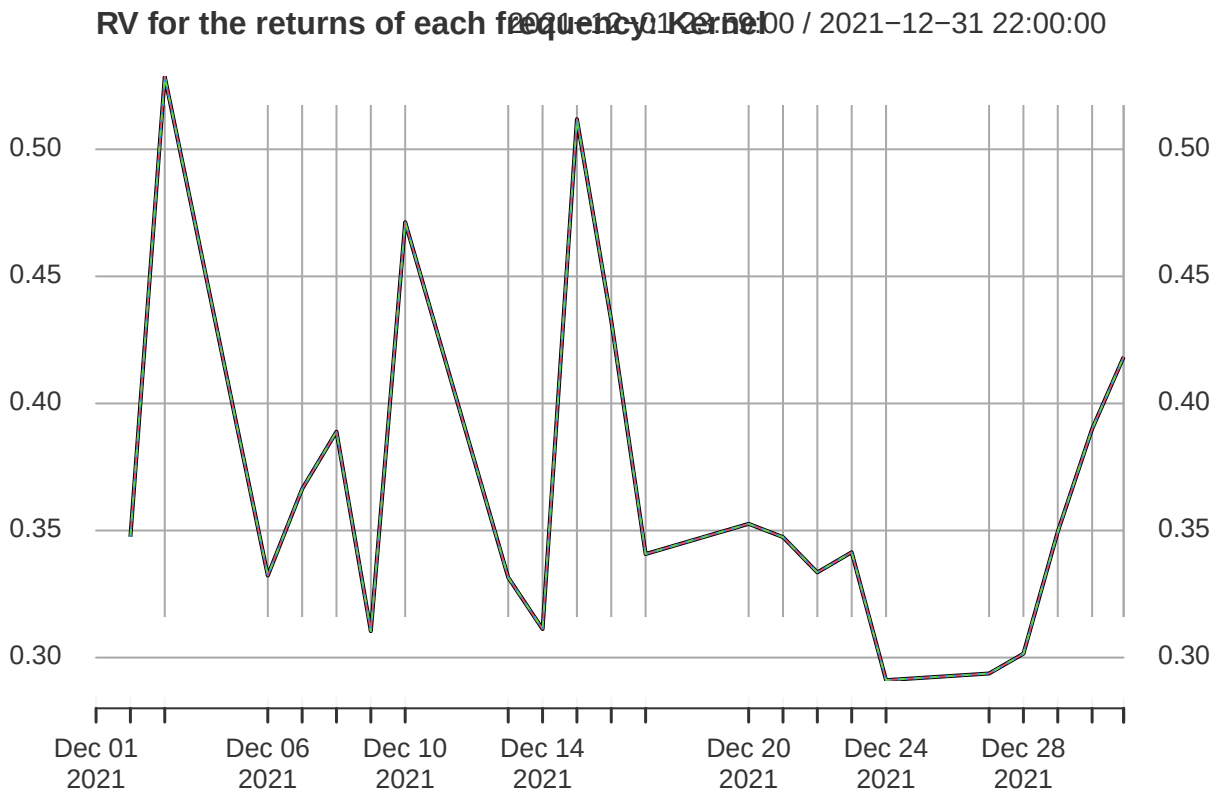

```
lines(rv1m_K^0.5, col = 2, lty = 1)
```




```
lines(rv5m_K^0.5, col = 3, lty = 2)
```

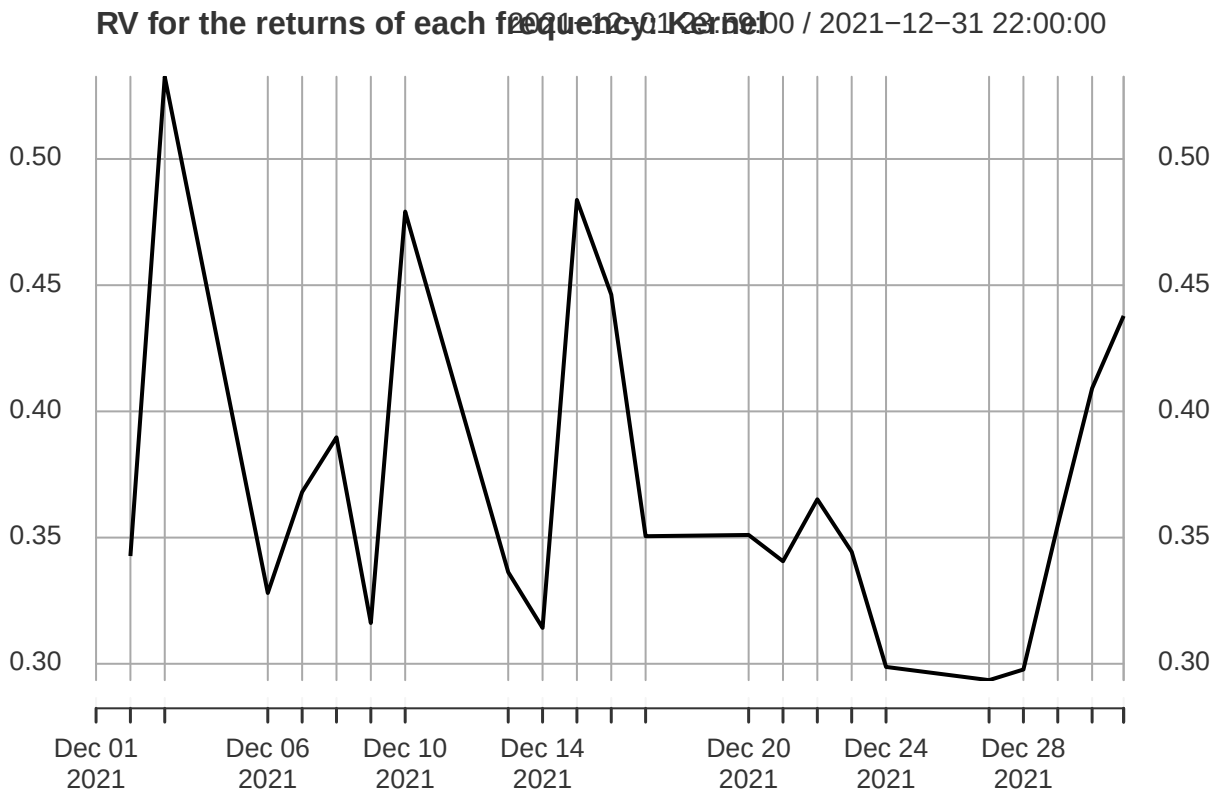



```
lines(rv10m_K^0.5, col = 4, lty = 3)
```

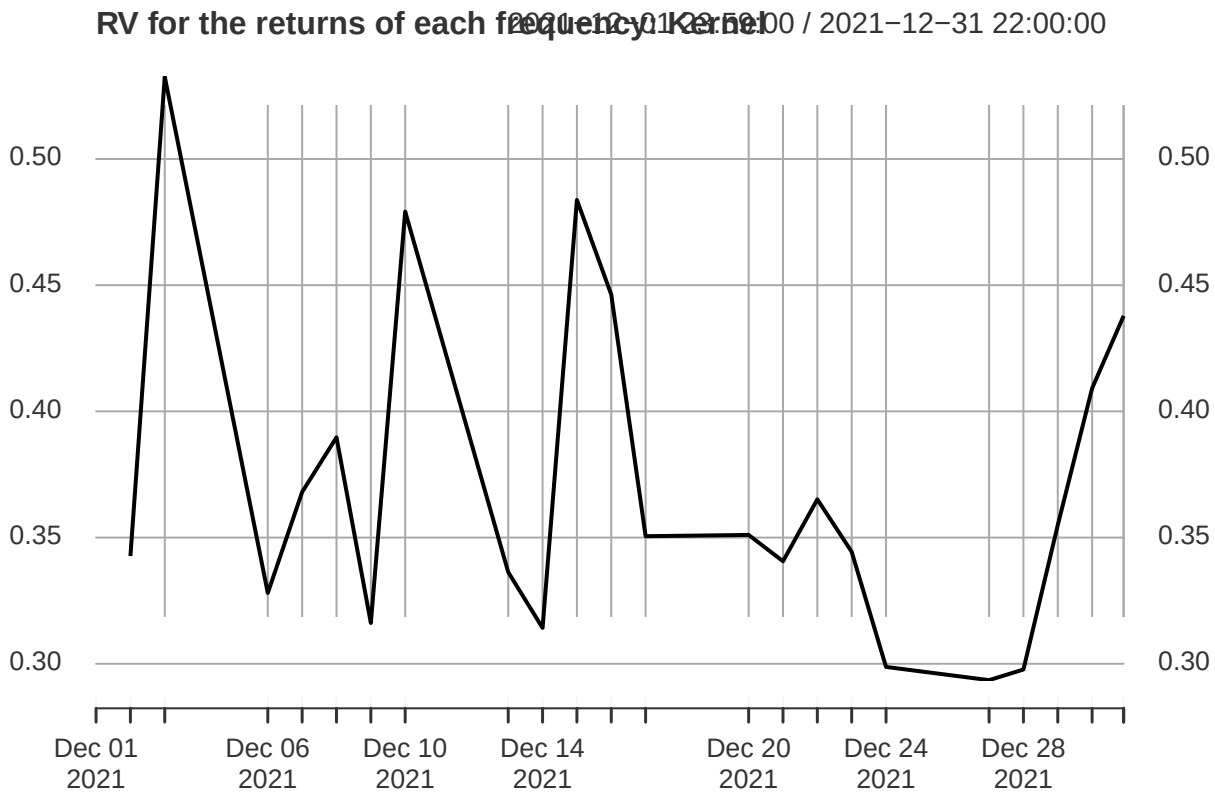


Averaging estimator:

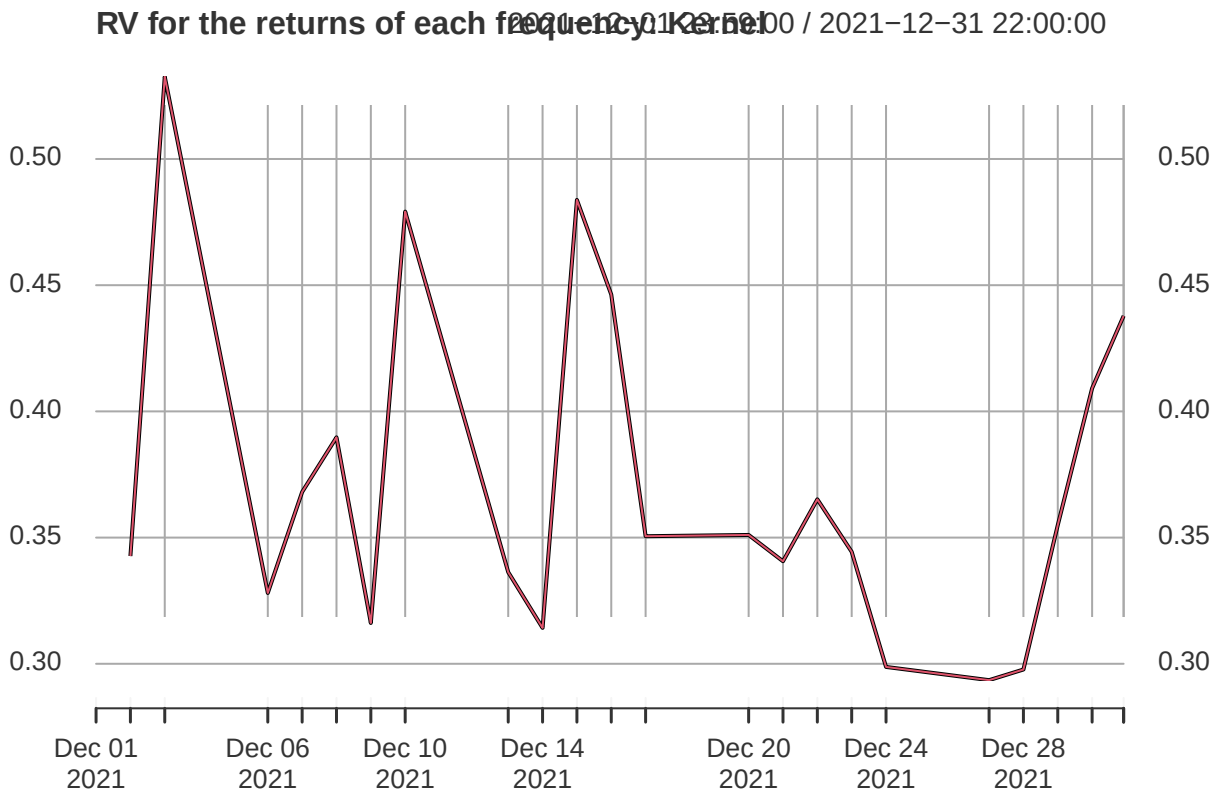
```
rv1s_A <- rAVGCov(ret1s, align.by="seconds", align.period=1)
rv10s_A <- rAVGCov(ret10s, align.by="seconds", align.period=10)
rv1m_A   <- rAVGCov(ret1m, align.by="minutes", align.period=1)
rv5m_A   <- rAVGCov(ret5m, align.by="minutes", align.period=5)
rv10m_A <- rAVGCov(ret10m, align.by="minutes", align.period=10)

plot(rv1s_A^0.5, main = "RV for the returns of each frequency: Kernel")
```



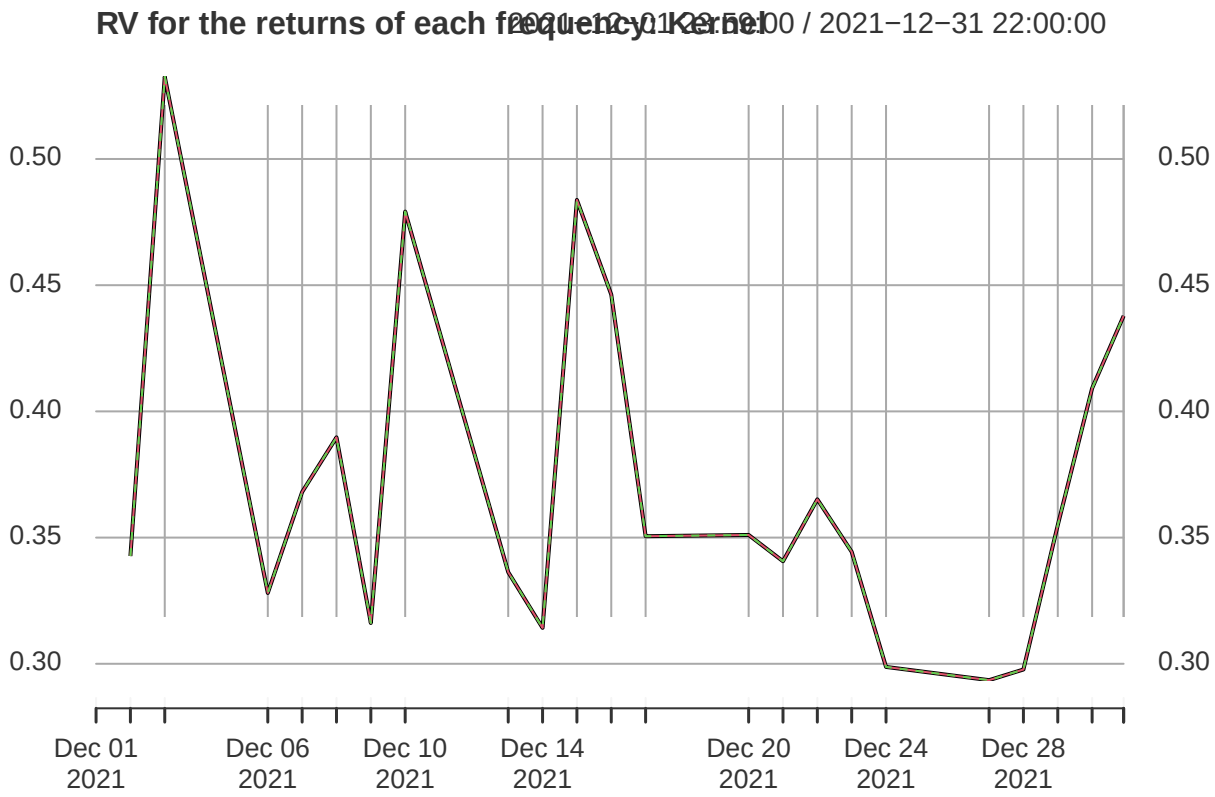

```
lines(rv10s_A^0.5, col = 1)
```



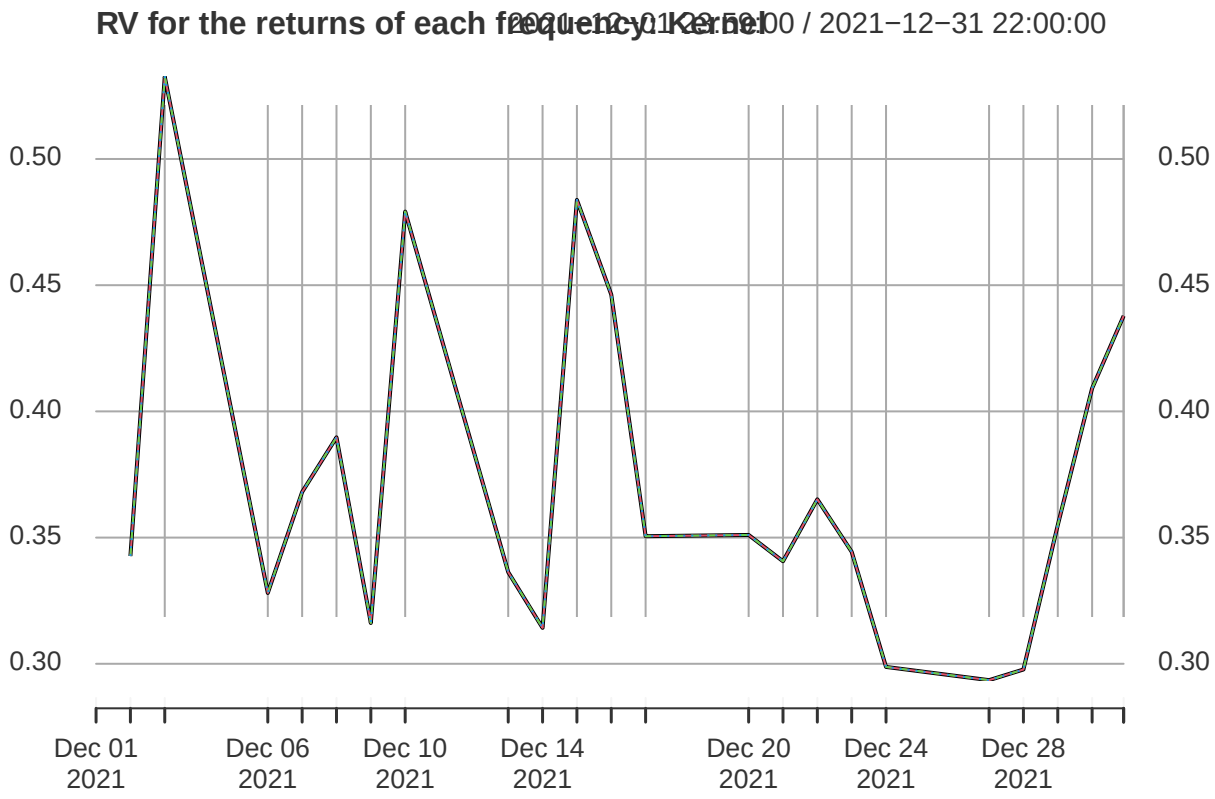

```
lines(rv1m_A^0.5, col = 2, lty = 1)
```




```
lines(rv5m_A^0.5, col = 3, lty = 2)
```




```
lines(rv10m_A^0.5, col = 4, lty = 3)
```



Two Time Scale Covariance Estimator and Pre-Averaing Estimator(using)

Note about Two Time Scale Covariance Estimator: the argument `pData` in `rTSCov`, and later in `rMSCov`, takes an `xts` object with the intraday price data, so the available dataset should be broken down by days:

```
ts1s_1 <- aggregateTS(eurus[,1], on="seconds", k=1)
ts1s_2 <- aggregateTS(eurus[,2], on="seconds", k=1)

index <- .indexwday(ts1s_1)
unique(index)
ts1s_1 <- ts1s_1[index %in% 1:5]

index <- .indexwday(ts1s_2)
unique(index)
ts1s_2 <- ts1s_2[index %in% 1:5]

# For WeekDay 1-22
# for (i in seq(1, 1440*22, by=1440)){
#   print(i)
# }

# WeekDay 1
a1 <- ts1s_1[1:1440,]
b1 <- ts1s_2[1:1440,]
rcovts1 <- rTSCov(pData = list(a1, b1), K = 50)
rcovmr1 <- rMRCov(pData = list(a1, b1), theta = 0.8)
```

```
# WeekDay 2
a2 <- ts1s_1[1441:2880,]
b2 <- ts1s_2[1441:2880,]
rcovts2 <- rTSCov(pData = list(a2, b2), K = 50)
rcovmr2 <- rMRCov(pData = list(a2, b2), theta = 0.8)
# WeekDay 3
a3 <- ts1s_1[2881:4320,]
b3 <- ts1s_2[2881:4320,]
rcovts3 <- rTSCov(pData = list(a3, b3), K = 50)
rcovmr3 <- rMRCov(pData = list(a3, b3), theta = 0.8)
# WeekDay 4
a4 <- ts1s_1[4321:5760,]
b4 <- ts1s_2[4321:5760,]
rcovts4 <- rTSCov(pData = list(a4, b4), K = 50)
rcovmr4 <- rMRCov(pData = list(a4, b4), theta = 0.8)
# WeekDay 5
a5 <- ts1s_1[5761:7200,]
b5 <- ts1s_2[5761:7200,]
rcovts5 <- rTSCov(pData = list(a5, b5), K = 50)
rcovmr5 <- rMRCov(pData = list(a5, b5), theta = 0.8)
# WeekDay 6
a6 <- ts1s_1[7201:8640,]
b6 <- ts1s_2[7201:8640,]
rcovts6 <- rTSCov(pData = list(a6, b6), K = 50)
rcovmr6 <- rMRCov(pData = list(a6, b6), theta = 0.8)
# WeekDay 7
a7 <- ts1s_1[8641:10080,]
b7 <- ts1s_2[8641:10080,]
rcovts7 <- rTSCov(pData = list(a7, b7), K = 50)
rcovmr7 <- rMRCov(pData = list(a7, b7), theta = 0.8)
# WeekDay 8
a8 <- ts1s_1[10081:11520,]
b8 <- ts1s_2[10081:11520,]
rcovts8 <- rTSCov(pData = list(a8, b8), K = 50)
rcovmr8 <- rMRCov(pData = list(a8, b8), theta = 0.8)
# WeekDay 9
a9 <- ts1s_1[11521:12960,]
b9 <- ts1s_2[11521:12960,]
rcovts9 <- rTSCov(pData = list(a9, b9), K = 50)
rcovmr9 <- rMRCov(pData = list(a9, b9), theta = 0.8)
# WeekDay 10
a10 <- ts1s_1[12961:14400,]
b10 <- ts1s_2[12961:14400,]
rcovts10 <- rTSCov(pData = list(a10, b10), K = 50)
rcovmr10 <- rMRCov(pData = list(a10, b10), theta = 0.8)
# WeekDay 11
a11 <- ts1s_1[14401:15840,]
b11 <- ts1s_2[14401:15840,]
rcovts11 <- rTSCov(pData = list(a11, b11), K = 50)
rcovmr11 <- rMRCov(pData = list(a11, b11), theta = 0.8)
# WeekDay 12
a12 <- ts1s_1[15841:17280,]
b12 <- ts1s_2[15841:17280,]
```

```
rcovts12 <- rTSCov(pData = list(a12, b12), K = 50)
rcovmr12 <- rMRCov(pData = list(a12, b12), theta = 0.8)
# WeekDay 13
a13 <- ts1s_1[17281:18720,]
b13 <- ts1s_2[17281:18720,]
rcovts13 <- rTSCov(pData = list(a13, b13), K = 50)
rcovmr13 <- rMRCov(pData = list(a13, b13), theta = 0.8)
# WeekDay 14
a14 <- ts1s_1[18721:20160,]
b14 <- ts1s_2[18721:20160,]
rcovts14 <- rTSCov(pData = list(a14, b14), K = 50)
rcovmr14 <- rMRCov(pData = list(a14, b14), theta = 0.8)
# WeekDay 15
a15 <- ts1s_1[20161:21600,]
b15 <- ts1s_2[20161:21600,]
rcovts15 <- rTSCov(pData = list(a15, b15), K = 50)
rcovmr15 <- rMRCov(pData = list(a15, b15), theta = 0.8)
# WeekDay 16
a16 <- ts1s_1[21601:23040,]
b16 <- ts1s_2[21601:23040,]
rcovts16 <- rTSCov(pData = list(a16, b16), K = 50)
rcovmr16 <- rMRCov(pData = list(a16, b16), theta = 0.8)
# WeekDay 17
a17 <- ts1s_1[23041:24480,]
b17 <- ts1s_2[23041:24480,]
rcovts17 <- rTSCov(pData = list(a17, b17), K = 50)
rcovmr17 <- rMRCov(pData = list(a17, b17), theta = 0.8)
# WeekDay 18
a18 <- ts1s_1[24481:25920,]
b18 <- ts1s_2[24481:25920,]
rcovts18 <- rTSCov(pData = list(a18, b18), K = 50)
rcovmr18 <- rMRCov(pData = list(a18, b18), theta = 0.8)
# WeekDay 19
a19 <- ts1s_1[25921:27360,]
b19 <- ts1s_2[25921:27360,]
rcovts19 <- rTSCov(pData = list(a19, b19), K = 50)
rcovmr19 <- rMRCov(pData = list(a19, b19), theta = 0.8)
# WeekDay 20
a20 <- ts1s_1[27361:28800,]
b20 <- ts1s_2[27361:28800,]
rcovts20 <- rTSCov(pData = list(a20, b20), K = 50)
rcovmr20 <- rMRCov(pData = list(a20, b20), theta = 0.8)
# WeekDay 21
a21 <- ts1s_1[28801:30240,]
b21 <- ts1s_2[28801:30240,]
rcovts21 <- rTSCov(pData = list(a21, b21), K = 50)
rcovmr21 <- rMRCov(pData = list(a21, b21), theta = 0.8)
# WeekDay 22
a22 <- ts1s_1[30241:31680,]
b22 <- ts1s_2[30241:31680,]
rcovts22 <- rTSCov(pData = list(a22, b22), K = 50)
rcovmr22 <- rMRCov(pData = list(a22, b22), theta = 0.8)
# For WeekDay 23:
```

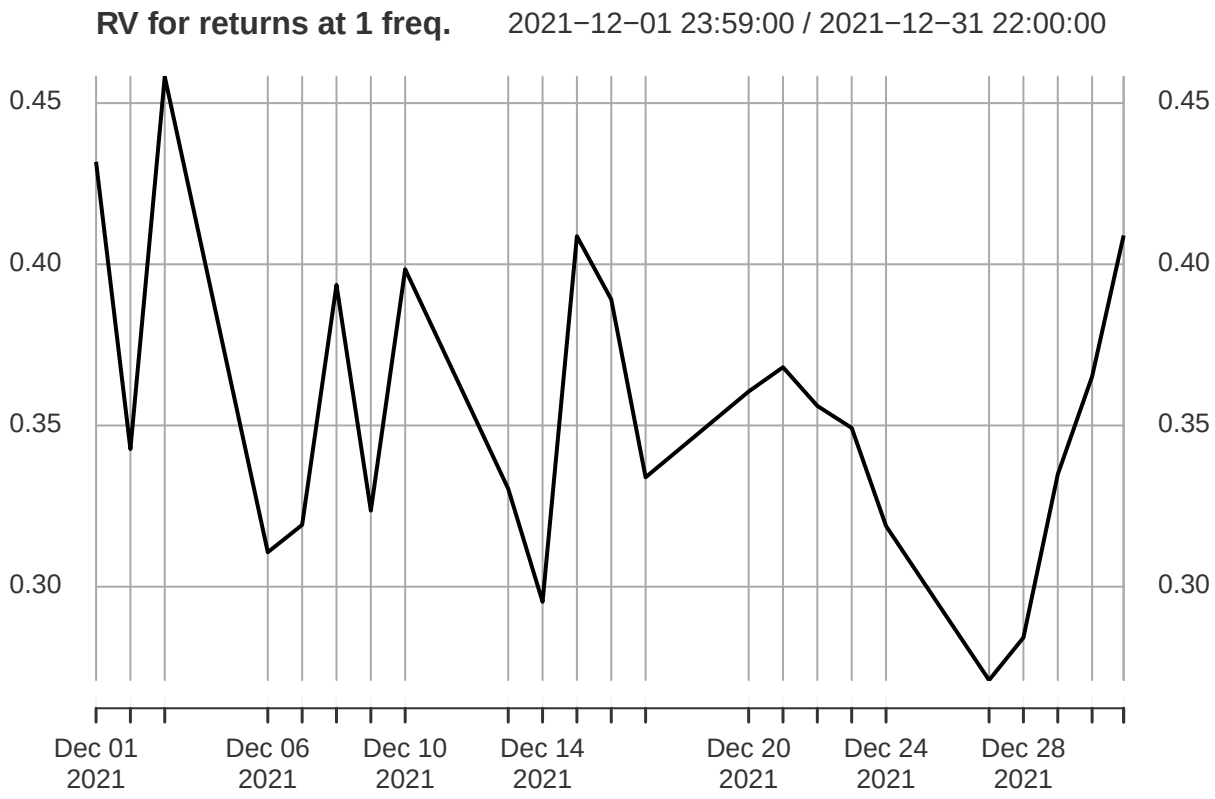
```
a23 <- ts1s_1[31681:33001,]
b23 <- ts1s_2[31681:33001,]
rcovts23 <- rTSCov(pData = list(a23, b23), K = 50)
rcovmr23 <- rMRCov(pData = list(a23, b23), theta = 0.8)
```

Conclusion:

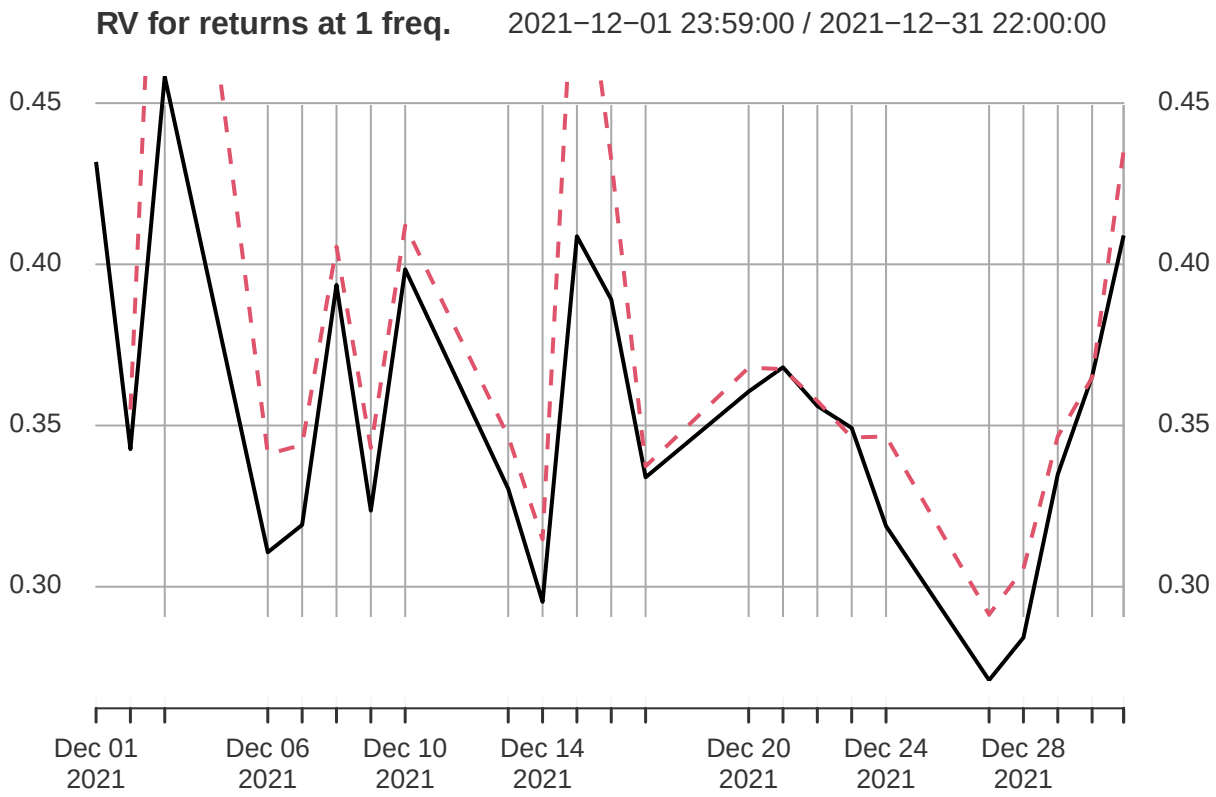
- 1 second aggregation in general outperforms other frequencies in terms of precision;
- Two Time Scales Estimator's performance is similar to the Pre-averaing estimator.

Moreover, the realized volatility measure is criticized because it is not robust to microstructure noise and to outliers of jumps, as it is shown in the previous graphs. An alternative to the estimator is the **median RV** estimator which consists of the taking the median of the intra-day returns:

```
medrv1s <- rMedRVar(ret1s, align.by="seconds", align.period=1)
plot(medrv1s^0.5, col=1, lty=1, main = "RV for returns at 1 freq.")
```




```
lines(rv1s^0.5, col=2, lty=2, lwd=2)
```


RV for returns at 1 freq.
2021−12−01 23:59:00 / 2021−12−31 22:00:00
0.45
0.40
0.35
0.30
Dec 01
2021
Dec 06
2021
Dec 10
2021
Dec 14
2021
Dec 20
2021
Dec 24
2021
Dec 28
2021

References

Heston, Steven L. 1993. "A closed-form solution for options with stochastic volatility with applications to bond and currency options." *The review of financial studies* 6(2):327–343.