Forecasting Realized Volatility on Intraday Bitcoin Returns with a Bayesain Approach

Jeff Gould, Sergio Leon

1 Introduction

Bitcoin has become one of the most talked about and controversial assets over the last several years. As of time of print, it trades at around $50,000 per coin, and has a total market value in excess of $1 trillion. One of the criticisms of bitcoin is that it can be an extremely volatile asset, sometimes dropping as much as 10% in a matter of hours. This is caused by the relative newness of the asset class. Any news that impacts, either negatively or positively, its global acceptance can drastically impact price volatility. Just recently, Elon Musk tweeted that Telsa would no longer be accepting bitcoin for vehicle purchases. This single tweet sent the price of bitcoin tumbling by 12%.

As bitcoin becomes more widely traded and makes up more of people's and institution's investment portfolios, being able to model and forecast that volatility is an important part of managing the risk in one's portfolio. Volatility has been studied extensively in equities markets, and is even tradable in the form of the CBOE's VIX. While volatility derivatives on bitcoin do not yet exist, the number of financial derivatives on cryptocurrencies is increasing, and it is not unreasonable to think it might be tradable in the near future. But in the interim, being able to forecast volatility is still an important part of portfolio management.

For the final project we aim to use Bayesian techniques to predict bitcoin volatility over a subsequent 24-hour period. Given that bitcoin volatility is influenced often by unpredictable events we want to establish a more convervative measure for evaluating the success of our models. For this reason we will evaluate model predictions as accurate if the actual volatility falls within the 100% confidence interval. If our models are able to achieve this for at least 50% of our test sample observations, we will consider our modeling effort a success. From a use case perspecitve, if this model is being used for portfolio risk management purposes being able to predict accurately within a 100% confidence interval can allow investors to reliably use the upper confidence limit as the highest volatility possible over a subsequent 24-hour period. For completeness, other more traditional metrics for model accuracy will also be provided.

Our dataset comes from the Coinbase Pro API, in the form of 5-minute Open-High-Low-Close-Volume (OHLCV) Candles, with date ranging from January 1, 2016 to April 16, 2021 (code to pull data available in Appendix B). One difficulty of bitcoin as opposed to traditional equiutes markets is that there is no central exchange. There are multiple places to trade bitcoin, both in the US and across the world. However, Coinbase is the largest exchange for bitcoin transactions, so it is still a sufficient source. We do need to make the assumption that the prices quoted on Coinbase are consistent with other exchanges, ie no arbitrage opportunities. There is also the issue of occasional server outages on Coinbase, causing "flash crashes" on bitcoin, while prices hold steady on other platforms. We somewhat deal with this issue by excluding trading days missing more than 40 minutes of trade data.

We calculate intraday variance as $\sum_{t=1}^{n} r_t^2$, where $r_t = \log(P_t) - \log(P_{t-1})$. Then Realized Volatility (RV) is simply the squareroot of the realized variance: $RV = \sqrt{\sum_{t=1}^{n} r_t^2}$. There are mutliple Volatility models that exist from reaserch in equities markets. Our goal is to explore a couple of these with both informative and non-informative priors, derive the parameters for the models, using data from January 2016 - June 30 2019, and then test the derived models and parameters on the data from July 1, 2019 - April 2021.

2 Methods

We explore three approaches to modeling the RV for bitcoin. First we took two approaches modeling the Heston Model. The Heston Model is one of the most recognized models for volatility in equities. The Heston Model is a mean-reverting stochastic process with the form of $dV_t = \theta(V_t - \omega)dt + \xi\sqrt{V_t}dB_t$, where ω is the long-run average volatility, xi is the variance, or "vol of vol", θ is the rate at which volatility reverts to the mean, and dB_t is standard Brownian Motion. In the general model, dB_t is correlated with the dW_t Brownian Motion for the change in price of the underlying asset (which is in the form of Geometric Brownian Motion), but since we aren't forecasting changes in price that is not necessary for our study. We tune the parameters to this model with two approaches, the first is to use flat/non-informative priors on the three parameters, and the other is using priors for ω centered at the mean and a prior for ξ centered at the variance. Our other approach is to tune parameters on $\log RV$, following an Ornstein-Uhlenbeck and Vasicek process (Gaussian Process): Define $V_t = \exp X_t$. Then $dX_t = \theta(\omega - X_t)dt + \xi dW_t$, and V_t follows a log-normal distribution. We hypothesize this transformation to be useful as volatility is a fat-tailed distribution, and the log-normal distribution typically approximates the underlying well

In the Heston Model, since dB_t is a standard Normal Random Variable, then $dV_t \sim \mathcal{N}(\theta(V_t - \omega), \xi^2 V_t)$. For the log-normal transformation, we follow the derivation from Tegner and Poulsen (2018): $X_{t+1}|X_t \sim \mathcal{N}\left(X_t e^{-\theta} + \omega(1 - e^{-\theta}), \frac{\xi^2(1-e^{-2\theta})}{2\theta}\right)$

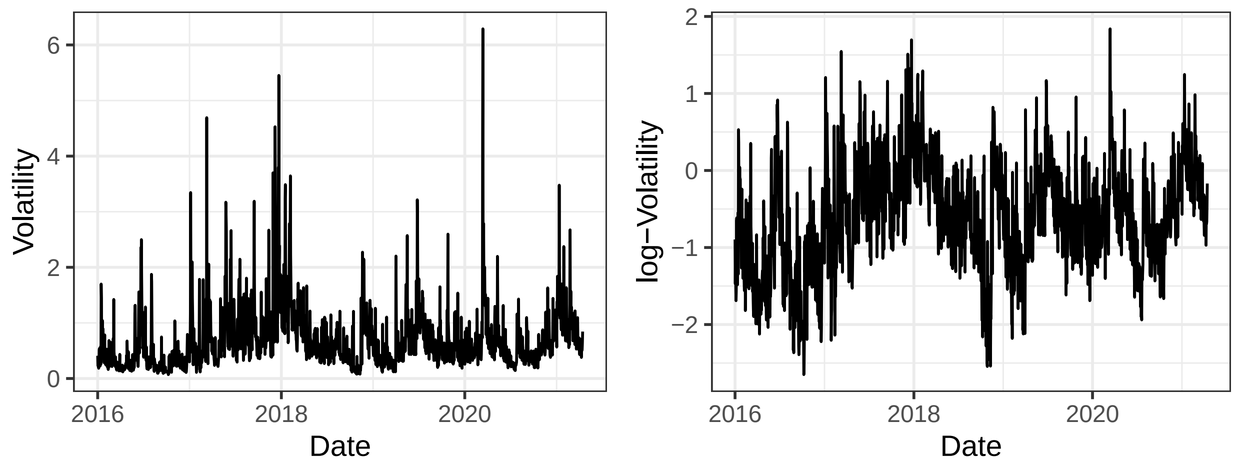

Figure 1: We plot the level volatility and log-volatility side-by-side, demonstrating the skew in the underlying distribution, and why sampling on the log-Vol may produce better results than on the level version.

For the Heston Model, we have a likelihood function of

$$\mathcal{L}(dV_t|V_t, \theta, \omega, \xi) \propto (\xi^2)^{-n/2} \exp\left[-\frac{1}{2\xi^2}\sum \frac{(dV_t - \theta(\omega - V_t))^2}{V_t}\right]$$

For ξ^2, we easily see that we get the kernel for an Inverse-Gamma distribution. Using the non-informative prior $\pi(\xi^2) \propto (\xi^2)^{-1}$, we get a posterior distribution of

$$P(\xi^2|V_t, dV_t, \theta, \omega) \sim IG\left(n/2, \sum \frac{(dV_t - \theta(\omega - V_t))^2)}{2V_t}\right)$$

There is no closed solution for θ or ω, so we use a MH step in our sampler. With a flat prior, these have the same posterior likelihood, the exponential term from our likelihood function:

$$P(\theta, \omega | \xi^2, dV_t, V_t) \propto \exp\left[-\frac{1}{2\xi^2}\sum \frac{(dV_t - \theta(\omega - V_t))^2}{V_t}\right]$$

Since ω is by definition the long-run mean of the volatility, we sample $omega^*$ from a normal distribution centered at $\bar{\omega}$, with the standard deviation tuned to improve our acceptance rate. Since θ is a rate parameter, $\theta < 1$, so we sample from a beta distribution for θ. Initially we start our run on the Uniform distribution to get an idea of where the true value is, and then we tune the shape and scale parameters to center around that point and improve our acceptance rate.

For our log-model, we again use the non-informative prior on ξ^2, $\pi(\xi^2) \propto (\xi^2)^{-1}$, with flat priors on θ and ω. This gives us the following likelihood function (See Appendix B for full derivation):

$$P(\xi^2, \theta, \omega | X) \propto (\xi^2)^{-1}\left[\frac{\theta^{n/2}}{(\xi^2(1 - e^{-2\theta}))^{n/2}}\right]^{-n/2} \exp\left[-\frac{\theta}{\xi^2(1 - e^{-2\theta})}\sum \left\{X_{t+1} - (X_t e^{-\theta} + \omega(1 - e^{-\theta}))\right\}^2\right]$$

Again, we have an Inverse Gamma posterior for ξ^2, with no closed form solutions for θ and ω. Instead we again use an MH step for θ and ω, using the same process as above

3 Results

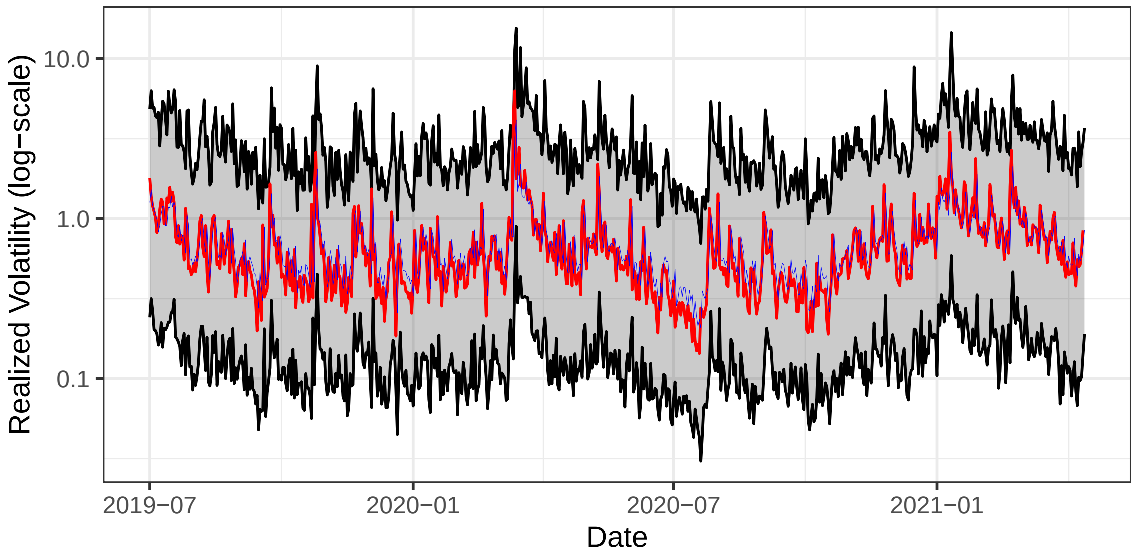

Figure 2: We plot the upper and lower bounds of our simulations using the results of the logRV model (black) with the actual Realized Volatility (red), and our point estimate (blue). Only three times (0.46%) did the actual realized volatility fall outside of our draws for realized volatility.

4 Discussion

5 References

6 Appendix

6.1 A

6.2 B Code

Here is the code for the Gibbs Sampler for Heston Model with non-informative priors:

```r
btc_candles <- read_csv("btcOpenCandles.csv") %>%
  mutate(dVt = lead(AnnualizedRV) - AnnualizedRV) #%>%
  # mutate(logAnnualRV = log(AnnualizedRV),
  #        dYt = lead(logAnnualRV) - logAnnualRV)

train <- btc_candles %>%
  filter(time < as.Date("2019-07-01"))

test <- btc_candles %>%
  filter(time >= as.Date("2019-07-01"))

Vt <- train$AnnualizedRV
dVt <- train$dVt

logL_omega <- function(Omega, Theta, Xi, .dVt = dVt, .Vt = Vt){
  -1 / (2 * Xi^2) * sum(((.dVt - Theta * (Omega - .Vt))^2) / .Vt)
}
logL_theta <- function(Theta, Omega, Xi,  .dVt = dVt, .Vt = Vt){
  -1 / (2 * Xi^2) * sum(((.dVt - Theta * (Omega - .Vt))^2) / .Vt)
}

burnIn <- 5000
thin <- 10
keep <- 1000
B <- burnIn + keep * thin

omega <- vector("numeric", B)
xi <- vector("numeric", B)
theta <- vector("numeric", B)

omega[1] = mean(Vt)
xi[1] = sd(Vt)^2
theta[1] = 0.25

n <- length(Vt)

omega_accept <- 0
theta_accept <- 0
set.seed(111)
for (b in 2:B) {

  phi_alpha <- n/2
  phi_beta <- sum((dVt - theta[b-1]*(omega[b-1] - Vt))^2 / (2*Vt))

  phi <- MCMCpack::rinvgamma(1, phi_alpha, phi_beta)
  xi[b] = sqrt(phi)

  U <- runif(1)
  ## MH for omega
  omega_star <- rnorm(1, mean = mean(Vt), sd = 0.5*sd(Vt))

  r <- exp(logL_omega(omega_star, theta[b-1], xi[b-1]) - logL_omega(omega[b-1], theta[b-1], xi[b-1]))
  if(U < min(r,1)){
    omega[b] = omega_star
    if(b > burnIn){
      omega_accept <- omega_accept + 1
    }
  }else{
    omega[b] = omega[b-1]
  }
  ## MH for theta
  theta_alpha = 4
  theta_beta = 16
  theta_star <- rbeta(1, theta_alpha, theta_beta)

  r <- exp(logL_theta(theta_star, omega[b-1], xi[b-1]) - logL_theta(theta[b-1], omega[b-1], xi[b-1])) *
    (dbeta(theta[b-1], theta_alpha, theta_beta) / dbeta(theta_star, theta_alpha, theta_beta))
  if(U < min(r,1)){
    theta[b] = theta_star
    if(b > burnIn){
      theta_accept <- theta_accept + 1
    }
  }else{
    theta[b] = theta[b-1]
  }

}

theta_accept / B
omega_accept / B

mean(omega[-c(1:burnIn)])
mean(Vt)

mean(theta[-c(1:burnIn)])
mean(xi[-c(1:burnIn)])
sd(Vt)

acf(theta[-c(1:(B/2))])
acf(xi[-c(1:(B/2))])
acf(omega[-c(1:(B/2))])

keepIdx <- seq(burnIn+1, B, by = thin)
ThetaKeep <- theta[keepIdx]
OmegaKeep <- omega[keepIdx]
XiKeep <- xi[keepIdx]

GibbsSamplerResults1 <- data.frame(omega = OmegaKeep,
                                    theta = ThetaKeep,
                                    xi = XiKeep)
saveRDS(GibbsSamplerResults1, "GibbsSamplerHeston1.rds")

plot(density(ThetaKeep))
hist(ThetaKeep)
mean(OmegaKeep)
median(OmegaKeep)

HestonModel <- function(dBt, theta, omega, Vt, xi){
  return(
    theta * (Vt - omega) + xi * sqrt(Vt) * dBt
  )
 }
# ItoHestonModel <- function(dBt, theta, omega, Vt, xi){
#   return(
#    (theta * (omega - Vt) - xi^2)/Vt + xi / sqrt(Vt) * dBt
#   )
# }

simDVT <- function(Vt_i, theta, omega, xi, date, dBt = NULL, sims = 1000){
  if(is.null(dBt)) dBt <- rnorm(sims)

  dYt <- t(sapply(dBt, HestonModel, Vt_i, theta, omega, xi)) %>% as.data.frame()
  dYt$Date = date
  return(dYt)

}


ParamsDf <- data.frame(theta = mean(ThetaKeep),
                       omega = mean(OmegaKeep),
                       xi = mean(XiKeep)) %>%
  expand_grid(test%>% select(time, AnnualizedRV)) %>%
  rename(date = time,
         Vt_i = AnnualizedRV)

testSim <- pmap_dfr(ParamsDf, simDVT)

checkResults <- testSim %>%
  pivot_longer(cols = V1:V1000, values_to = "simDVt") %>%
  select(-name) %>%
  left_join(test %>% select(Date = time, dVt, AnnualizedRV) %>% mutate(Vt1 = lead(AnnualizedRV))) %>%
  mutate(Vt1_pred = AnnualizedRV + simDVt)

checkResults %>%
  group_by(Date) %>%
  mutate(upper95 = quantile(Vt1_pred, 0.95),
         lower95 = quantile(Vt1_pred, 0.05)) %>%
  mutate(upper = max(Vt1_pred),
         lower = min(Vt1_pred),
         pointEst = mean(Vt1_pred)) %>%
  select(Date, upper95, lower95, Vt1) %>%
  distinct() %>%
  mutate(in95 = (Vt1 >=lower95 & Vt1 <= upper95)) %$% mean(in95, na.rm = T)

checkResults %>%
  select(Date, Vt1_pred, Vt1) %>%
  ungroup() %>%
  group_by(Date, Vt1) %>%
  summarise(pointEst = mean(Vt1_pred)) %$% cor(Vt1, pointEst, use = "complete.obs")^2

GibbsSamplerResultsHeston <- read_rds("GibbsSamplerHeston1.rds")

WAIC <- test %>%
  select(Date, dVt, AnnualizedRV) %>%
  mutate(Vt_1 = dVt + AnnualizedRV,
         i = row_number()) %>%
  expand_grid(GibbsSamplerResultsHeston%>% mutate(b = 1:keep)) %>%
  mutate(mu = AnnualizedRV + theta*(omega - AnnualizedRV),
         sigma2 = xi*sqrt(AnnualizedRV)) %>%
  mutate(Ly_i = dnorm(Vt_1, mu, sqrt(sigma2)))

lppd <- WAIC %>% group_by(i) %>% summarise(BB = 1/n() * sum(Ly_i)) %$% sum(log(BB), na.rm = T)

pWAIC <-  WAIC %>%
  group_by(i) %>%
  mutate(meanLogl = 1/keep * sum(log(Ly_i))) %>%
  summarise(inside = sum((log(Ly_i) - meanLogl)^2)) %$% sum((1/(keep-1)) * inside, na.rm = T)

WAIC_return <- -2 * lppd + 2 * pWAIC
```

Code for the Heston Model with informative priors:

```
library(tidyverse)

omega <- 0.633
theta <- 0.327
xi <- 0.514


btc_candles <- read_csv("btcOpenCandles.csv") %>%
  mutate(dVt = lead(AnnualizedRV) - AnnualizedRV)

test <- btc_candles %>%
  filter(time >= as.Date("2019-07-01"))
train <- btc_candles %>%
  filter(time < as.Date("2019-07-01"))

Vt <- train$AnnualizedRV
dVt <- train$dVt


omega_func <- function(Omega, Theta, Xi, q, r, .dVt = dVt, .Vt = Vt){
  exp(-1/2*Xi^(2)*sum(((.dVt - Theta * (Omega - .Vt))^2) / .Vt)-q*Omega)*Omega^(r-1)
}
```

```r
#logL_theta <- function(Theta, Omega, Xi,  .dVt = dVt, .Vt = Vt){
# -1 / (2 * Xi^2) * sum(((.dVt - Theta * (Omega - .Vt))^2) / .Vt)
#}

burnIn <- 5000
thin <- 10
keep <- 1000
B <- burnIn + keep * thin

omega <- vector("numeric", B)
xi <- vector("numeric", B)
theta <- vector("numeric", B)

omega[1] = mean(Vt)
xi[1] = sd(Vt)^2
theta[1] = 0.20

n <- length(Vt)
a<-4
d<-1
s<-3
t<-2
tune<-0.13

omega_accept <- 0
theta_accept <- 0
set.seed(111)
for (b in 2:B) {


  ##Gibbs for xi
  phi_alpha <- (n/2)+a
  phi_beta <- (sum(((dVt - theta[b-1]*(omega[b-1] - Vt))^(2))/(Vt))-2*d)/2

  phi <- MCMCpack::rinvgamma(1, phi_alpha, phi_beta)
  xi[b] = sqrt(phi)

  U <- runif(1)

  #Gibbs for theta
  mu<- sum((dVt*(omega[b-1]-Vt)))/sum((omega[b-1]-Vt)^(2))
  sd<-sqrt(xi[b-1]^(2)*(sum((omega[b-1]-Vt)^(2)/Vt)^(-1)))

  theta_new<- rnorm(1,mean=mu,sd=sd)
  theta[b] = theta_new

  ## MA for omega
  omega_star <- rnorm(1, mean = mean(Vt), sd = tune*sd(Vt))

  r <- omega_func(omega_star, theta[b-1], xi[b-1],s,t)/omega_func(omega[b-1], theta[b-1], xi[b-1],s,t)

  if(U < min(r,1)){
    omega[b] = omega_star
    if(b > burnIn){
      omega_accept <- omega_accept + 1
    }
  }else{
    omega[b] = omega[b-1]
  }

}



HestonModel <- function(dBt, theta, omega, Vt, xi){
  return(
    theta * (Vt - omega) + xi * sqrt(Vt) * dBt
  )
}


simDVT <- function(Vt_i, theta, omega, xi, date, dBt = NULL, sims = 1000){
  if(is.null(dBt)) dBt <- rnorm(sims)

  dYt <- t(sapply(dBt, HestonModel, Vt_i, theta, omega, xi)) %>% as.data.frame()
  dYt$Date = date
  return(dYt)

}


ParamsDf <- data.frame(theta = theta,
                       omega =omega,
                       xi = xi) %>%
  expand_grid(test%>% select(time, AnnualizedRV)) %>%
  rename(date = time,
         Vt_i = AnnualizedRV)

set.seed(111)
testSim <- pmap_dfr(ParamsDf, simDVT)

checkResults <- testSim %>%
  pivot_longer(cols = V1:V1000, values_to = "simDVt") %>%
  select(-name) %>%
  left_join(test %>% select(Date = time, dVt, AnnualizedRV) %>% mutate(Vt1 = lead(AnnualizedRV))) %>%
  mutate(Vt1_pred = AnnualizedRV + simDVt)

write_rds(checkResults, "HestonModel2Sim.rds")

checkResults %>%
  group_by(Date) %>%
  mutate(upper95 = quantile(Vt1_pred, 0.95),
         lower95 = quantile(Vt1_pred, 0.05)) %>%
  mutate(upper = max(Vt1_pred),
         lower = min(Vt1_pred),
         pointEst = mean(Vt1_pred)) %>%
  select(Date, upper95, lower95, Vt1, Vt1_pred) %>%
  distinct() %>%
  mutate(in95 = (Vt1 >=lower95 & Vt1 <= upper95)) %$%mean(in95, na.rm = T) #%$% cor(Vt1_pred, Vt1, use

checkResults %>%
  select(Date, Vt1_pred, Vt1) %>%
  ungroup() %>%
  group_by(Date, Vt1) %>%
  summarise(pointEst = mean(Vt1_pred)) %$% cor(Vt1, pointEst, use = "complete.obs")^2

### WAIC
keepIdx <- seq(burnIn+1, B, by = thin)
ThetaKeep <- theta[keepIdx]
OmegaKeep <- omega[keepIdx]
XiKeep <- xi[keepIdx]

GibbsSamplerResults2 <- data.frame(omega = OmegaKeep,
                                   theta = ThetaKeep,
                                   xi = XiKeep)

saveRDS(GibbsSamplerResults2, "GibbsSamplerHeston2.rds")

GibbsSamplerResultsHeston <- read_rds("GibbsSamplerHeston2.rds")

WAIC <- test %>%
  select(Date, dVt, AnnualizedRV) %>%
  mutate(Vt_1 = dVt + AnnualizedRV,
         i = row_number()) %>%
  expand_grid(GibbsSamplerResultsHeston%>% mutate(b = 1:keep)) %>%
  mutate(mu = AnnualizedRV + theta*(omega - AnnualizedRV),
         sigma2 = xi*sqrt(AnnualizedRV)) %>%
  mutate(Ly_i = dnorm(Vt_1, mu, sqrt(sigma2)))

lppd <- WAIC %>% group_by(i) %>% summarise(BB = 1/n() * sum(Ly_i)) %$% sum(log(BB), na.rm = T)

pWAIC <-  WAIC %>%
  group_by(i) %>%
  mutate(meanLogl = 1/keep * sum(log(Ly_i))) %>%
  summarise(inside = sum((log(Ly_i) - meanLogl)^2)) %$% sum((1/(keep-1)) * inside, na.rm = T)

WAIC_return <- -2 * lppd + 2 * pWAIC
```

And lastly, the code for the log-RV model:

```
btc_candles <- read_csv("btcOpenCandles.csv") %>%
  mutate(dVt = lead(AnnualizedRV) - AnnualizedRV) %>%
  mutate(logAnnualRV = log(AnnualizedRV),
        dYt = lead(logAnnualRV) - logAnnualRV) %>%
  select(Date,time,  AnnualizedRV, dVt, logAnnualRV, dYt)

train <- btc_candles %>%
  filter(time < as.Date("2019-07-01"))
```

```r
test <- btc_candles %>%
  filter(time >= as.Date("2019-07-01"))


Xt <- train$logAnnualRV
#dYt <- train$dYt

## Make a function for the kernel in the exponent for easier use later
expKernel <- function(k, eps, theta, X = Xt){
  - k / (eps * ( 1 - exp(-2 * k))) * sum(
    (lead(X) - (X * exp(-k) + theta * (1 - exp(-k))))^2
  , na.rm = T)
}

logLKappa <- function(k, eps, theta, X = Xt){
  n/2  * log(k) - n/2 * log(1 - exp(-2*k)) + expKernel(k, eps, theta, X)
}


burnIn <- 10000
thin <- 10
keep <- 1000
B <- burnIn + keep * thin

theta <- vector("numeric", B)
eta2 <- vector("numeric", B)
kappa <- vector("numeric", B)

theta[1] = mean(Xt)
eta2[1] = sd(Xt)^2
kappa[1] = 0.25

n <- length(Xt)

theta_accept <- 0
kappa_accept <- 0

set.seed(111)
for (b in 2:B) {

  U <- runif(1)

  ## Gibbs Sample eta2
  eta2[b] <- MCMCpack::rinvgamma(1, n/2, -expKernel(kappa[b-1], 1, theta[b-1]))


  ## MH for theta
  theta_star <- rnorm(1, mean(Xt), sd(Xt)/2)

  r <- exp(expKernel(kappa[b-1], eta2[b-1], theta_star) - expKernel(kappa[b-1], eta2[b-1], theta[b-1]))

  if(U < min(r,1)){
    theta[b] = theta_star
    if(b > burnIn){
      theta_accept <- theta_accept + 1
    }
  }else{
    theta[b] = theta[b-1]
  }

  ## MH for eta
  kappa_alpha = 3
  kappa_beta = 9
  kappa_star <- rbeta(1, kappa_alpha, kappa_beta)
  #kappa_star <- runif(1)

  r <- exp(logLKappa(kappa_star, eta2[b-1], theta[b-1]) - logLKappa(kappa[b-1], eta2[b-1], theta[b-1]))
    (dbeta(kappa[b-1], kappa_alpha, kappa_beta) / dbeta(kappa_star, kappa_alpha, kappa_beta))
  if(U < min(r,1)){
    kappa[b] = kappa_star
    if(b > burnIn){
      kappa_accept <- kappa_accept + 1
    }
  }else{
    kappa[b] = kappa[b-1]
  }

}


kappa_accept / B
theta_accept / B

mean(theta[-c(1:burnIn)])
mean(Xt)

mean(kappa[-c(1:burnIn)])
mean(eta2[-c(1:burnIn)]) %>% sqrt()
sd(Xt)

acf(theta[-c(1:(B/2))])
acf(kappa[-c(1:(B/2))])
acf(eta2[-c(1:(B/2))])

keepIdx <- seq(burnIn+1, B, by = thin)
ThetaKeep <- theta[keepIdx]
KappaKeep <- kappa[keepIdx]
eta2Keep <- eta2[keepIdx]

GibbsSamplerResultsLog <- data.frame(omega = ThetaKeep,
                                     theta = KappaKeep,
                                     xi = sqrt(eta2Keep))
write_rds(GibbsSamplerResultsLog, "GibbsSamplerLog.rds")
##calcWAIC
WAIC <- test %>%
  select(Date, logAnnualRV) %>%
  mutate(Vt_1 = lead(logAnnualRV),
         i = row_number()) %>%
  expand_grid(GibbsSamplerResultsLog%>% mutate(b = 1:keep)) %>%
  mutate(mu = logAnnualRV * exp(-theta) + omega * (1 - exp(-theta)),
         sigma2 = (xi^2 * (1 - exp(-2*theta))) / (2*theta)) %>%
  mutate(Ly_i = dnorm(Vt_1, mu, sqrt(sigma2)))

lppd <- WAIC %>% group_by(i) %>% summarise(BB = 1/n() * sum(Ly_i)) %$% sum(log(BB), na.rm = T)

pWAIC <-  WAIC %>%
  group_by(i) %>%
  mutate(meanLogl = 1/keep * sum(log(Ly_i))) %>%
  summarise(inside = sum((log(Ly_i) - meanLogl)^2)) %$% sum((1/(keep-1)) * inside, na.rm = T)

WAIC_return <- -2 * lppd + 2 * pWAIC




plot(density(ThetaKeep))
hist(KappaKeep)
hist(eta2Keep)


calcXt1 <- function(Xt, k, eps2, theta, Date, n = 1000){
  mu <- Xt * exp(-k) + theta * (1 - exp(-k))
  var <- (eps2 * ( 1 - exp(-2*k))) / (2*k)
  Xt1 <- rnorm(n, mu, sqrt(var))
  simXt1 <- expand_grid(Date, Xt1) %>%
    mutate(Vt1 = exp(Xt1))
  return(simXt1)
}

K <- mean(KappaKeep)
Eps2  <- mean(eta2Keep)
Theta <- mean(ThetaKeep)

ParamsDf <- data.frame(k = K,
                       eps2 = Eps2,
                       theta = Theta) %>%
  expand_grid(test%>% select(Date, logAnnualRV)) %>%
  rename(Xt = logAnnualRV)

testSim <- pmap_dfr(ParamsDf, calcXt1)

testSim %>%
  rename(Xt1_pred = Xt1,
         Vt1_pred = Vt1) %>%
  group_by(Date) %>%
  mutate(upper95 = quantile(Vt1_pred, 0.95),
         lower95 = quantile(Vt1_pred, 0.05)) %>%
  mutate(upper = max(Vt1_pred),
         lower = min(Vt1_pred),
         pointEst = mean(Vt1_pred)) %>%
  select(Date, upper, lower, pointEst, upper95, lower95) %>%
  distinct() %>%
  left_join(test %>% select(Date, AnnualizedRV, logAnnualRV)) %>%
  ungroup() %>%
  mutate(Xt1 = lead(logAnnualRV),
         Vt1 = lead(AnnualizedRV)) %>%
  mutate(inCI = case_when(Vt1 <= upper95 & Vt1 >= lower95 ~ 1,
                          TRUE ~ 0)) %>% write_rds("logSimResults.rds")
  #filter(lubridate::day(Date) %in% c(1,6,11,16,21,26)) %>%
  # ggplot(aes(x = AnnualizedRV)) +
  # geom_point(aes(y = pointEst), col = "red") +
  #geom_point(aes(y = dVt), col = "blue")
  ggplot(aes(x = Date)) +
  geom_line(aes(y = upper), color = "black") +
  geom_line(aes(y = lower), color = "black") +
  geom_line(aes(y = Vt1), color = "red") +
  #scale_y_log10() +
  theme_bw()
  #geom_errorbar(aes(ymin = lower, ymax = upper)) +
  geom_point(aes(y = Vt1), col = "red")  +
  geom_point(aes(y = pointEst), col = "blue") +
  theme_bw()
```