Homework 4: solution suggestions

Paul Huebner

April 30, 2020

Question 1

part 1

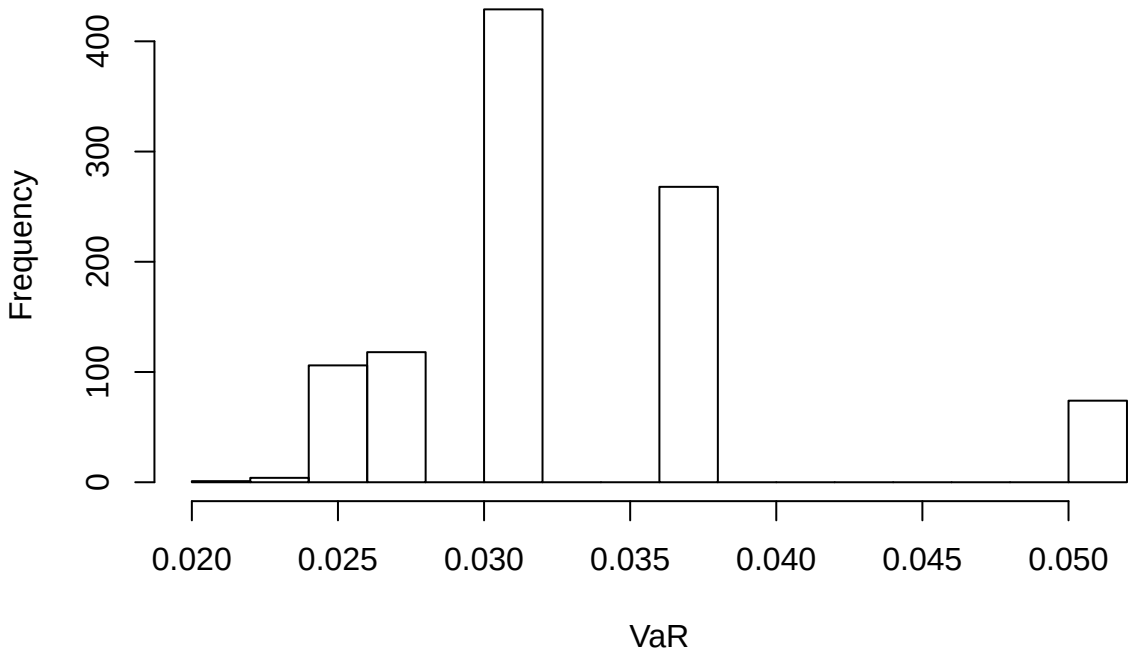
```
data <- read.csv('hw3_returns2.csv')
data$Date <- as.Date(data$Date, "%m/%d/%Y")

comp_dates <- data %>% filter(Date >= as.Date('2015-01-01')) %>% .$Date
```

Below I show how to bootstrap using the historical method for the first day of 2015. The method works exactly the same for every day afterwards, it will just get more computanionally expensive. Alternatively, you can come up with smart ideas how to preserve draws and decrease computation time[1].

Note that the key of using the bootstrap method is to sample *with replacement*.

```
start_length <- 1000 - length(comp_dates)


# bootstrap
set.seed(1)
bootdata <- sample(data[1:start_length,2], size = 1000*start_length, replace = TRUE)
bootsample <-
  data.frame(index = rep(1:1000, each = start_length),
             return = bootdata)

VaRs <-
bootsample %>%
  group_by(index) %>%
  arrange(return) %>%
  summarize(VaR = return[ceiling(0.01*start_length)])

hist(-VaRs$VaR, xlab = 'VaR', main = '')
```

[1] E.g. you draw $1,000^2$ time from a standard uniform distribution, and define days as evenly spaced intervals on $[0,1]$. You can re-use draws from the uniform distribution throughout the history of the VaR, with only the spacing changing. This is a valid approach to get a VaR at any given day, but be cautious about the inference you would draw about the evolution of the confidence interval over time.




```
quantile(-VaRs$VaR,c(0.025,0.975))
```

```
##       2.5%      97.5%
## 0.02550368 0.05060242
```

Blindly resampling the exponentially weighted VaR is not very useful because the whole idea of this approach is that the data-generating process changes over time, which is reflected in higher weights for more recent observations. However, you can mimick the idea by resampling using exponential weights (the sample function in R has an argument for probability weights), and then proceed as above with the historical. However, this conceptually isn't a bootstrap for exponential-weighted VaR, but instead a bootstrap for historical VaR where bootstrap samples are generated according to exponential weights.

Another way to do this that is closer to an actual bootstrap for exponential-weighted VaR would resample like with the historical method, but retaining the original weights (and reusing them) in the calculation of VaR. This still isn't perfect as weights would in general are not unique and do not sum to 1 any more. In any case, the key point here is to realize that even something seemingly straightforward like a bootstrap can easily become conceptually challenging.

For a parametric method, we can use the estimate for standard errors on the slides. For the first day of 2015 we get the following s.e. (it's straightforward to do this for every day in the sample).

```
parameters <-
  data %>%
    filter(Date < as.Date('2015-01-01')) %>%
    summarise(mu = mean(Return),
              sigma = sqrt(sum((Return-mu)^2)/n()),
              N = n()) %>%
    as.numeric
```

```r
x <- parameters[1] - parameters[2]*qnorm(0.01)
se <- sqrt(0.99*0.01/parameters[3])/dnorm(x, parameters[1], parameters[2])
se
```

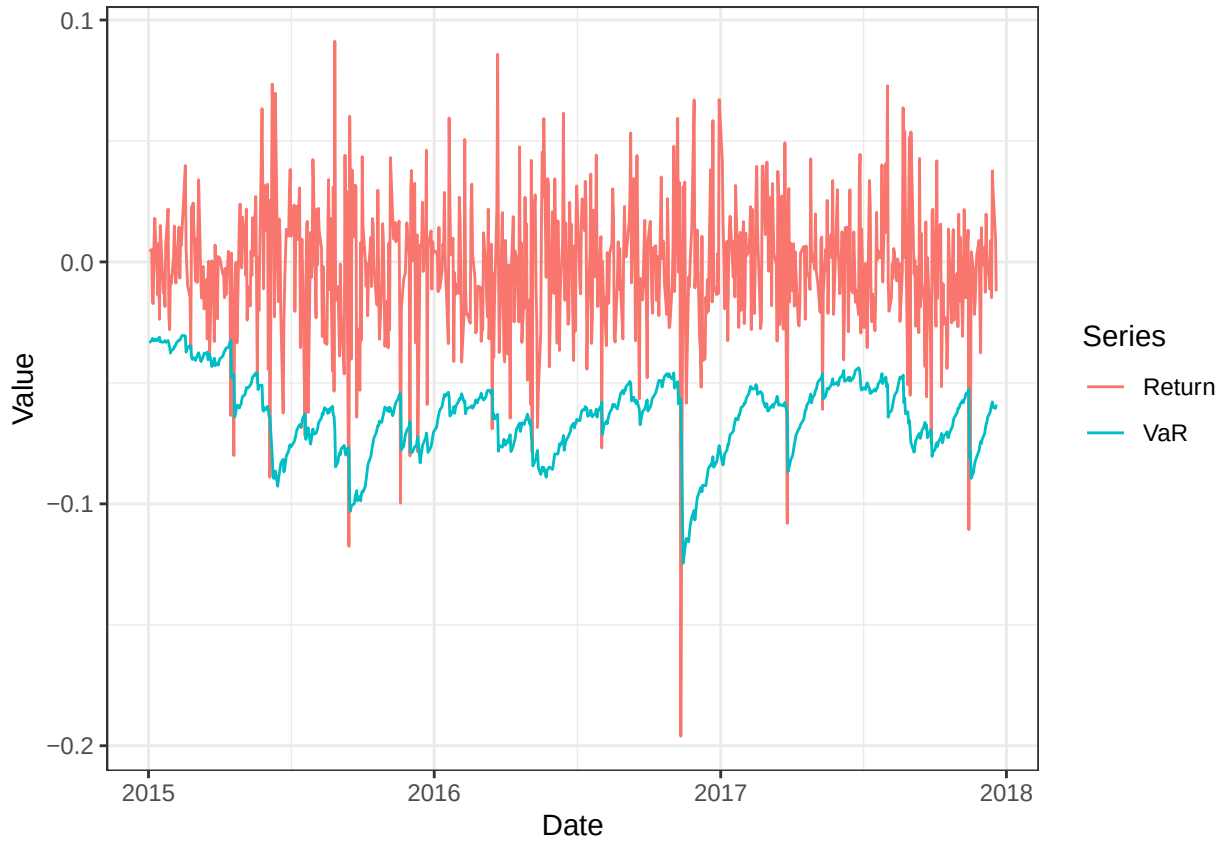
```
## [1] 0.002973418
```

Based on the standard error, you can compute a 95% confidence interval through $[VaR - 1.96 \times s.e., VaR + 1.96 \times s.e.]$.

part 2

There is a typo in the question, λ should be 0.94 here.

```r
# code from class
n = nrow(data)

# sigma_t^2 = lambda * sigma_{t-1}^2 + (1-lambda) * R_{t-1}^2

lambda = 0.94

sigma_2_ewma = rep(0., n)
# initialize at a reasonable value: first squared returns or a prior
sigma_2_ewma[1] = data$Return[1]^2
# sigma_2_ewma[1] = 0.01

# use the updating formula
for (t in 2:n){
  sigma_2_ewma[t] = lambda * sigma_2_ewma[t-1] + (1- lambda) * data$Return[t-1]^2
}
sigma_2_ewma[1] = NA

sigma_ewma = sqrt(sigma_2_ewma)
worst_case_ewma = qnorm(0.01) * sigma_ewma
VaR_ewma = - worst_case_ewma

plotdata <- data.frame(data,
                  VaR = worst_case_ewma) %>% filter(Date %in% comp_dates)
plotdata <- gather(plotdata, Series, Value, Return:VaR)
ggplot(plotdata, aes(x = Date, y = Value)) + geom_line(aes(col = Series)) + theme_bw()
```



part 3

The following code uses ML to estimate the parameters of a GARCH(1,1) using the first year of data.

```
returns <- data  %>% filter(!(Date %in% comp_dates)) %>% select(Return) %>% unlist

variances <- function(rets, omega, alpha, beta){
  sigma2 <- numeric(length(rets))
  sigma2[1] <- omega/(1 - alpha - beta)
  for(t in 2:length(rets)){
    sigma2[t] <- omega + alpha*rets[t-1]^2 + beta*sigma2[t-1]
  }
  sigma2
}

logLik <- function(par, rets = returns){
  omega <- par[1]
  alpha <- par[2]
  beta <- par[3]
  sigma2 <- variances(rets, omega, alpha, beta)
  -sum(-log(sigma2) - rets^2/sigma2)
}

parameters <- optim(c(0.01, 0.3, 0.3), logLik)$par
```

```r
optim(c(0.01, 0.3, 0.3), logLik, control = list(reltol = 1e-14))
```

```
## $par
## [1] 1.239034e-05 9.321343e-02 8.295707e-01
##
## $value
## [1] -1962.665
##
## $counts
## function gradient
##      416       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Convergence 0 shows successful convergence in optim. The VaR after one year of data is:

```r
-sqrt(tail(variances(returns, parameters[1], parameters[2], parameters[3]), 1)) * qnorm(0.01)
```

```
## [1] 0.03275308
```