Risk Model

Austin Griffith

November 21, 2017

Risk Management

This code models the risk of a randomly selected set of 100 firms stocks. The time period over which the firms are analyzed were for 2005 to 2010, and compared with values determined from 2000 to 2010.

The SAS code used to pull the data from the DSF data file will be provided. The data cleaning using SAS was stopped once the 100 firms were selected for each set of years. For the risk models, historical data was used to determine a starting point. As seen in the SAS code, the historical data was used from the prior year (in this case, 2004 and 1999).

Importing the Data

For the first step, the data was imported from the csv files. These csv's had the returns, date and firm permanent number for the random 100 firms pulled from the dsf files.

It was assumed that there was $1 million invested into each firm, leading to the assumption in the read file that there were equal weighted position in each firm. Therefore, a simple average of returns was used.

The data importing step was created using a function that takes a file name and reads the data in. This simplified the importing process for multiple files.

```
# set working directory
# needs to be adjusted when on new computer
setwd("C:/Users/Austin/GIT_profile/risk_management")
getwd()
```

```
## [1] "C:/Users/Austin/GIT_profile/risk_management"
```

```
# import data
# returns are in %
# sum return percentages by day
# assumed equal investment weight in each firm
read_func <- function(value, I)
{
  filename = paste(value, ".csv", sep="")
  ret = read.csv(filename, header=TRUE)

  firms = ret[["PERMNO"]]
  count = length(unique(firms))
  port = count*I

  ret_sum = aggregate(.~DATE, data=ret, FUN=sum)
  ret_sum = ret_sum[order(as.Date(ret_sum$DATE, "%m/%d/%Y"), decreasing=FALSE),]
  ret_date = unique(ret_sum[["DATE"]])

  keeps = c("RET","DATE")
  ret_sum = ret_sum[keeps]
```

```
  # gets average
  i = 1
  while(i < nrow(ret_sum))
  {
    ret_sum[i,"RET"] = ret_sum[i,"RET"]/count
    i = i + 1
  }

  data = list("portfolio" = port, "returns" = ret_sum)
  return(data)
}

invest = 1000000 # investment per firm
file1 = "returns_main"
file2 = "returns_comp"
file3 = "returns_main_hist"
file4 = "returns_comp_hist"

# reads files, gets list of returns and value
data_m = read_func(file1,invest)
data_c = read_func(file2,invest)
data_mh = read_func(file3,invest)
data_ch = read_func(file4,invest)
```

Histogram of Returns

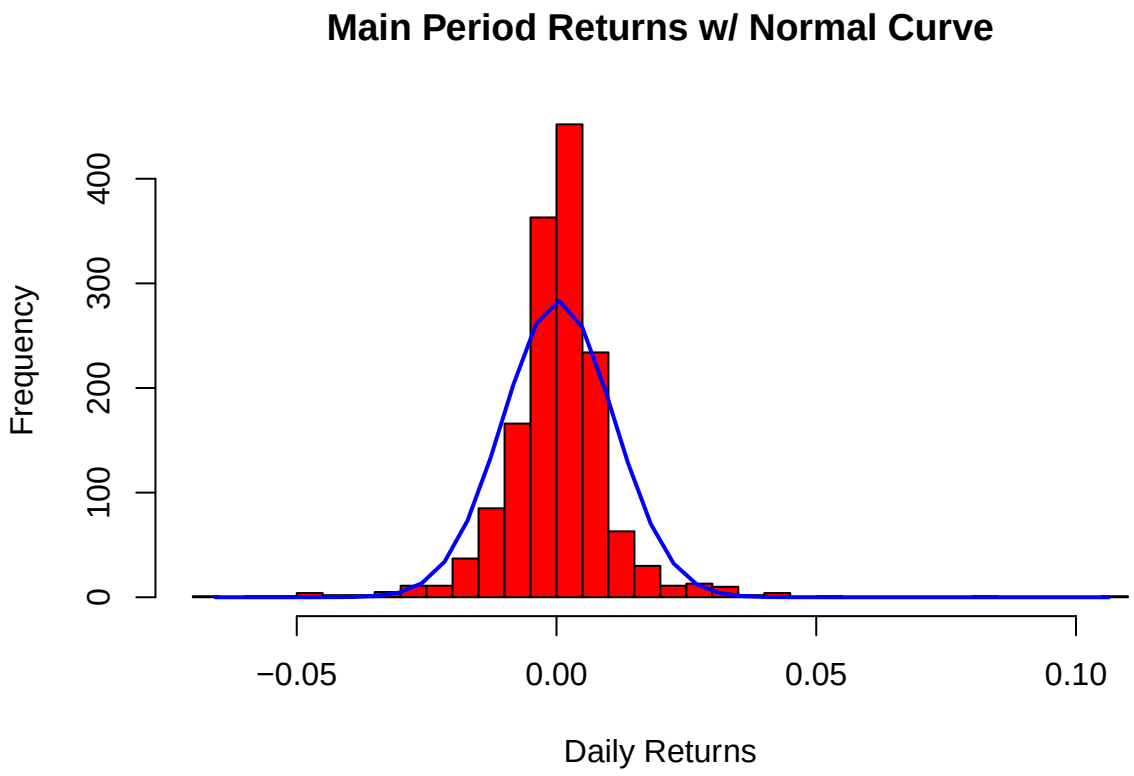
Once the data was pulled from the csv's, returns were graphed to view the historical distributions with a normal distribution overlay. This was used to test that the return data was imported properly, and view the normality of the returns.

The histogram also served as a sanity check for the VaR. If the VaR value looked unreasonable (too large, for example), it could be quickly checked against the distribution of returns.

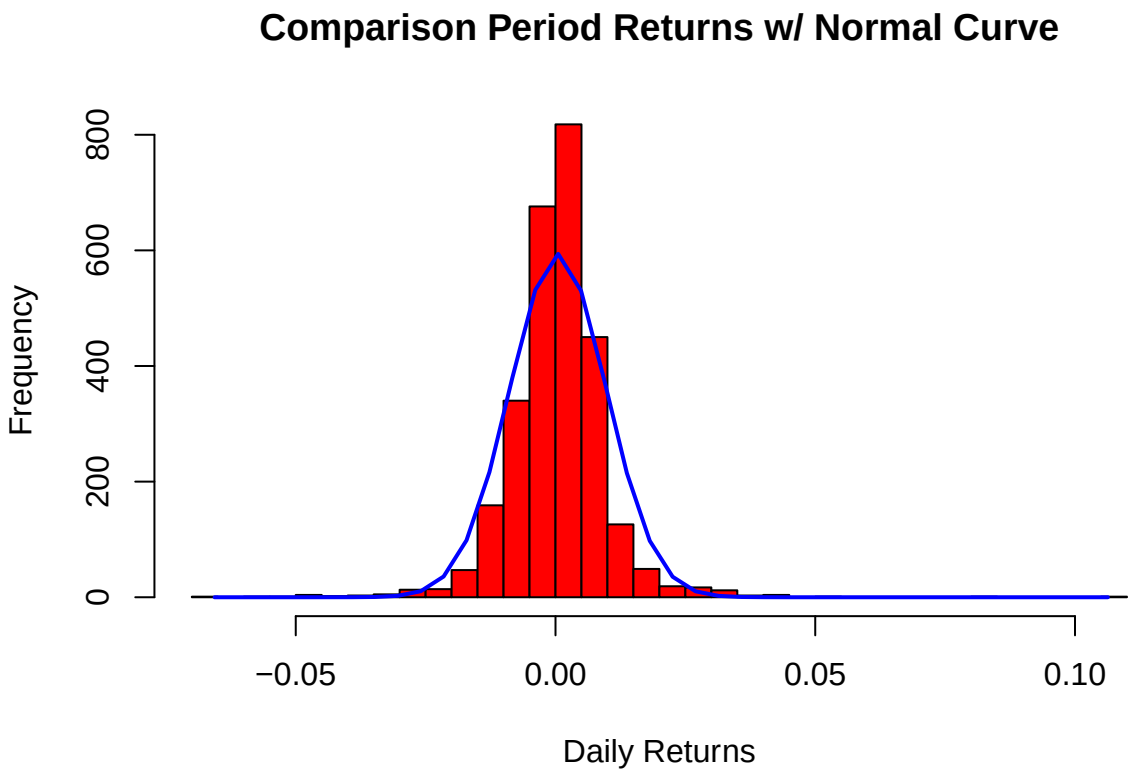
```
# histogram function with normal dist overlay
vec_histogram <- function(x, names)
{
  h = hist(x, breaks=(length(x)/50), col="red", xlab=names$xlabel,
           main=names$title)
  xfit = seq(min(x),max(x),length=40)
  yfit = dnorm(xfit, mean=mean(x), sd=sd(x))
  yfit = yfit*diff(h$mids[1:2])*length(x)
  lines(xfit, yfit, col="blue", lwd=2)
}

# lists with labels for histograms
hist_label_m = list("title" = "Main Period Returns w/ Normal Curve",
                    "xlabel" = "Daily Returns")
hist_label_c = list("title" = "Comparison Period Returns w/ Normal Curve",
                    "xlabel" = "Daily Returns")

# write histograms for historical returns
vec_histogram(data_m$returns[["RET"]], hist_label_m)
```




```
vec_histogram(data_c$returns[["RET"]], hist_label_c)
```



VaR, $VaR and Expected Shortfall

VaR was the first of the calculations used to determine the risk of the randomly chosen firms. The $VaR was calculated simultaneously in the same function. Both of these risk metrics require a confidence interval with which to measure the downside risk. For this particular case, 95% confidence was chosen.

The function also determines the expected shortfall of the positions over the time period. The value is conditional on distribution of the normal variable being below the VaR. This can be used to determine the magnitude of the worst case scenario of the positions held in the firm over the time period.

```
#function to calculate var
var_calc <- function(returns,port,a)
{
  r_vec = returns[["RET"]]
  vol = sd(r_vec)
  cumdist = qnorm(1-a,0,1)

  var = abs(quantile(r_vec,1-a))
  dol_var = abs(quantile(r_vec,1-a)*port)
  exp_short = vol*dnorm(cumdist)/(1-a)

  data = list("VaR"=var, "$VaR"=dol_var,
              "Expected_Shortfall" = exp_short)
  return(data)
}
```

```r
# confidence interval for var
conf = 0.95

# var calculations
var_m = var_calc(data_m$returns,data_m$portfolio,conf)
var_c = var_calc(data_c$returns,data_c$portfolio,conf)

# prints out var Variables
var_m
```

```
## $VaR
##         5%
## 0.01484422
##
## $`$VaR`
##      5%
## 1484422
##
## $Expected_Shortfall
## [1] 0.02193938
```

```r
var_c
```

```
## $VaR
##         5%
## 0.01260002
##
## $`$VaR`
##      5%
## 1260002
##
## $Expected_Shortfall
## [1] 0.01916717
```

The values outputed by the function give insight into the two time periods. For one, the VaR for 2005-2010 was 1.4%; 2 tenths of a percent larger than that of the 2000-2010 VaR. This implies a greater volatility for the shorter time period. Since both of these samples capture the entirety of the 2008 Financial Crisis, it should be expected that the the variance of the 2005 sample is impacted more due to the shorter time period over which it takes place.

Similarly, the $VaR is greater for the 2005 period. The $VaR is simply the VaR scaled with the size of the investment. Since the investment is identical between both periods, the $VaR is influenced by the same forces as the VaR.

The expected shortfall (ES) for each time period shows how the lower bound tails of the returns behave. ES for the 2005 period was 2.2%. This tells us that the loss on a given day within the time period will be 2.2%, given the value of a loss on that is greater than the VaR.

The ES for the comparison period is 1.9%. That makes the difference between VaR and ES for each period less than 1%. The relatively small difference between values shows that the returns over the period were not so extreme as to make a large jump in losses should the VaR be passed. This is reinforced by the distribution returns shown above. There are few outliers on the lower bound tails.

Daily Models

Both of the functions below capture the day by day swing of the variance, VaR and ES of the periods.

The first function is the calculation of the RiskMetrics model (also known as the "exponential smoother"). This solves for the variance of the next day using todays variance and returns. The coefficient of the model is assumed to be 0.94 by the RiskMetrics model.

The function uses the initial variance estimate as the variance across all assets for the previous ten days. In order to find the previous ten days, a historical data set was imported (shown in code above).

```
# risk metrics one day modeling
oneday_f <- function(returns,port,hist_returns,a)
{
  # initial variance using historical data
  # variance based off of previous 10 days
  hist = hist_returns[["RET"]]
  variance_0 = var(hist[length(hist)-10:length(hist)])

  # variables for while loop
  lamda = 0.94
  i = 1
  variance = c(0)
  var = c(0)
  exp_short= c(0)
  cumdist = qnorm(1-a,0,1)

  while(i <= nrow(returns))
  {
    variance_1 = lamda*variance_0 + (1-lamda)*((returns[i,"RET"])^2)
    variance[i] = variance_1
    var[i] = -1*sqrt(variance_1)*cumdist
    exp_short[i] = sqrt(variance_1)*dnorm(cumdist)/(1-a)

    variance_0 = variance_1
    i = i + 1
  }
  returns$m_variance = variance
  returns$VaR = var
  returns$ExpShort = exp_short
  return(returns)
}
```

The second function uses the GARCH model. The function is structurally similar in its estimation of the future variance, but adds a drift constant. The function also uses a GARCH library command to solve for the omega, alpha and beta constants. It also outputs the summary statistics and coefficient values.

The function also uses a historical data set to determine the initial variance of the model.

```
# garch model
garch_f <- function(returns,hist_returns,a)
{
  # initial variance using historical data
  # variance based off of previous 10 days
  hist = hist_returns[["RET"]]
  variance_0 = var(hist[length(hist)-10:length(hist)])

  # function to solve for garch model variables
  x.g = garchFit(~garch(1,1),returns[["RET"]])
  summary(x.g)
```

```
  coef(x.g)

  # variables for loop and garch model
  i = 1
  variance = c(0)
  var = c(0)
  exp_short= c(0)
  cumdist = qnorm(1-a,0,1)
  alpha = coef(x.g)[3]
  beta = coef(x.g)[4]
  omega = coef(x.g)[2]

  while(i <= nrow(returns))
  {
    variance_1 = omega + beta*variance_0 + alpha*((returns[i,"RET"])^2)
    variance[i] = variance_1
    var[i] = -1*sqrt(variance_1)*cumdist
    exp_short[i] = sqrt(variance_1)*dnorm(cumdist)/(1-a)

    variance_0 = variance_1
    i = i + 1
  }
  returns$VaR = var
  returns$ExpShort = exp_short
  returns$g_variance = variance
  return(returns)
}
```