Seminar 7

In this class we will:

1. Implement GARCH VaR
2. Analyze and compare VaR forecasts
3. Perform backtesting with Violation Ratios
4. Implement multivariate EWMA and HS VaR
5. Stress testing

Loading packages

We first load the packages for this seminar:

In [2]:
```
library(lubridate)
library(rugarch)
```

We will continue building VaR forecasts for the returns of Microsoft. You should load the `VaR` data frame from the previous seminar, which should contain the columns:

- HS300
- HS1000
- HS2000
- EWMA_VaR

In [3]:
```
# Load files from the previous seminar
load("VaR.RData")

# Load returns
load("Y.RData")
y <- Y$MSFT
dates <- Y$date

# Parameters
portfolio_value = 1000
p = 0.05
```

GARCH VaR

To implement VaR forecast using GARCH for a given estimation window, we need to recursively fit the specified GARCH model to the moving estimation window to get the optimal parameters, and use these to forecast conditional volatility for the following day. Once we have the conditional volatility, we estimate VaR using the inverse cdf for the specified probability and the portfolio value. For example, if the estimation window is 1000 days, we should:

1. Use data from day 1 to day 1000 to fit the GARCH model
2. Extract the optimal coefficients
3. Find $\hat{\sigma}_{t+1}$ using the GARCH equation
4. Compute VaR by: $\text{VaR}_{t+1}(p) = -\hat{\sigma}_{t+1} F_R^{-1}(p)\,\vartheta$

Note that we will be fitting thousands of GARCH models, so this will take a long time to run.

We want to forecast GARCH VaR using estimations windows of 300 and 2000 days. The best way to do so is to write a function:

In [ ]:
```
# Steps inside the function:
# Take as argument
# GARCH spec, here the default
# Probability, here 0.05
# Portfolio value, here 1000
# Estimation window, here 1000
spec <- ugarchspec()
p <- 0.05
portfolio_value <- 1000
WE <- 1000

# Determine number of observations
n <- length(y)

# Initialize empty VaR vector

VaR <- rep(NA, n)

# Do a loop for the forecast
for (i in 1:(n-WE)){

    # Subset the dataset to the estimation window
    window <- y[i:(i+WE-1)]

    # Fit the GARCH
    res <- ugarchfit(spec = spec, data = window, solver = "hybrid")

    # Save coefficients
    omega <- coef(res)['omega']
    alpha <- coef(res)['alpha1']
    beta <- coef(res)['beta1']

    # Estimate sigma2 using the last observation of window
    sigma2 <- omega + alpha*tail(window,1)^2 + beta*tail(res@fit$var,1)

    # Allocate the VaR forecast in the vector
    VaR[i+WE] <- -sqrt(sigma2) * qnorm(probability) * portfolio_value
}
```

Now let's put everything inside a function:

In [ ]:
```
# Function that creates a GARCH forecast

DoGARCH <- function(y, spec, probability = 0.05, portfolio_value = 1, WE = 1000){
    # GARCH function that takes as argument:
    # y: A vector of returns, ordered by date
    # spec: The ugarchspec object with the GARCH specification
    # probability: The probability to be used for VaR - Default 5%
    # portfolio_value: The portfolio value - Default 1
    # WE: Estimation window for the forecast - Default 1000 days

    # To calculate elapsed time, first get the current time
    old <- Sys.time()

    # Print message
    cat("Doing GARCH VaR forecast", "\n",
        "Estimation window:", WE, "\n",
        "Number of observations:", length(y), "\n",
        "VaR probability:", probability, "\n",
        "Portfolio value:", portfolio_value)

    # Number of observations
    n <- length(y)

    # Initialize empty VaR vector
    VaR <- rep(NA, n)

    # Do a loop for the forecast
    for (i in 1:(n-WE)){

        # Subset the dataset to the estimation window
        window <- y[i:(i+WE-1)]

        # Fit the GARCH
        res <- ugarchfit(spec = spec, data = window, solver = "hybrid")

        # Save coefficients
        omega <- coef(res)['omega']
        alpha <- coef(res)['alpha1']
        beta <- coef(res)['beta1']

        # Estimate sigma2 using the last observation of window
        sigma2 <- omega + alpha*tail(window,1)^2 + beta*tail(res@fit$var,1)

        # Allocate the VaR forecast in the vector
        VaR[i+WE] <- -sqrt(sigma2) * qnorm(probability) * portfolio_value
    }

    # Get the new time and print the elapsed time
    time <- difftime(Sys.time(), old, units = "secs")
    cat("\n", "Elapsed time:", round(time,4), "seconds")

    # Return the VaR vector
    return(VaR)
}
```

Doing the GARCH VaR forecast for estimation windows of 300 and 2000 days:

In [5]:
```
# Create specification
spec <- ugarchspec(
    variance.model = list(garchOrder= c(1,1)),
    mean.model= list(armaOrder = c(0,0), include.mean=FALSE)
)
```

In [6]:
```
# GARCH VaR for 300 days
GARCH300 <- DoGARCH(y, spec = spec, probability = 0.05, portfolio_value = 1000, WE = 300)
```

```
Doing GARCH VaR forecast
 Estimation window: 300
 Number of observations: 7559
 VaR probability: 0.05
 Portfolio value: 1000
 Elapsed time: 374.7003 seconds
```

Since the code takes a long time to run, we can save the output to avoid having to run it again:

In [7]:
```
# Saving the output
save(GARCH300, file = "GARCH300.RData")
```

In [8]:
```
# GARCH VaR for 2000 days
GARCH2000 <- DoGARCH(y, spec = spec, probability = 0.05, portfolio_value = 1000, WE = 2000)
```

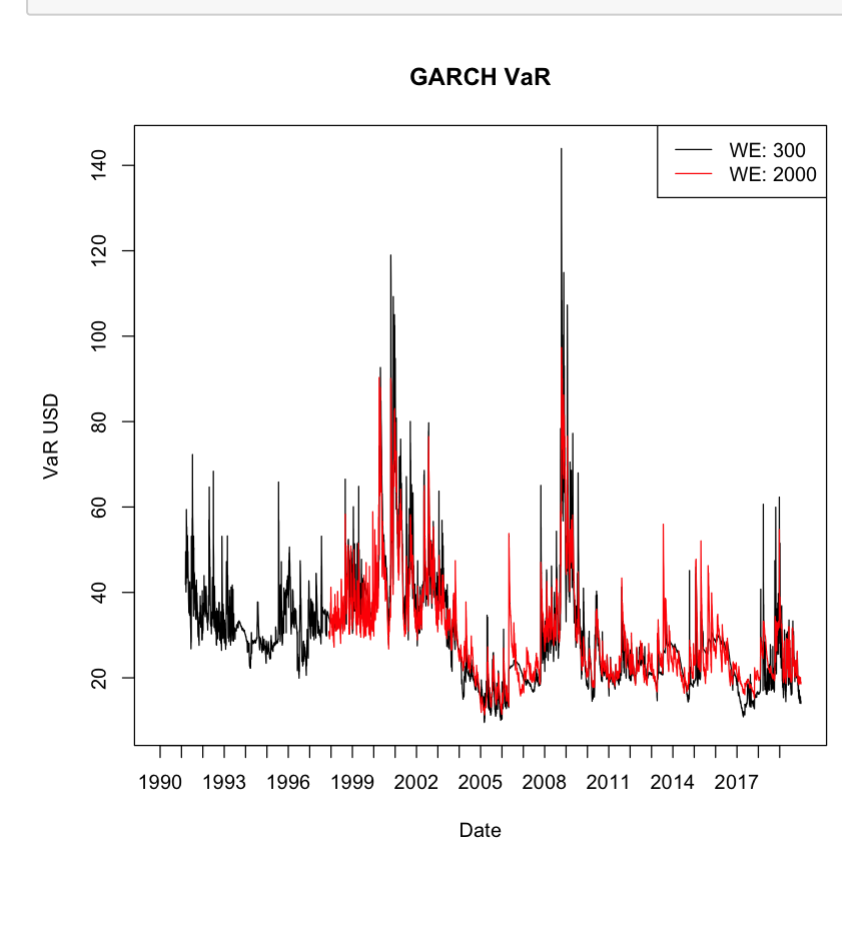
```
Doing GARCH VaR forecast
 Estimation window: 2000
 Number of observations: 7559
 VaR probability: 0.05
 Portfolio value: 1000
 Elapsed time: 631.4514 seconds
```

In [9]:
```
# Saving the output
save(GARCH2000, file = "GARCH2000.RData")
```

In [10]:
```
# If we have it already saved, we can load it by:
load("GARCH300.RData")
load("GARCH2000.RData")
```

We can plot them together:

In [60]:
```
# Bind into a matrix
GARCH_VaR <- cbind(GARCH300, GARCH2000)

# Plot and modify axis to include dates
matplot(dates, GARCH_VaR, type = "l", lty = 1, col = 1:2, xaxt = "n", main = "GARCH VaR", xlab = "Date"
, ylab = "VaR USD")
axis.Date(1, at = seq(min(dates), max(dates), by = "years"))

# Legend
legend("topright", legend = c("WE: 300", "WE: 2000"), lty = 1, col = 1:2)
```



Analyze and compare VaR forecasts

We now have 6 different VaR forecasts:

1. EWMA using WE = 30 days
2. HS using WE = 300 days
3. HS using WE = 1000 days
4. HS using WE = 2000 days
5. GARCH using WE = 300 days
6. GARCH using WE = 2000 days

In order to analyze and compare them, let's put them together in a matrix:

In [61]:
```
# Creating a matrix for all VaR forecasts
VaR <- cbind(VaR, GARCH300, GARCH2000)
```

First, let's compare the means and standard deviation for each forecast. To get the standard deviation of each column, we can use the `apply()` , which takes as input a matrix, a function to be applied to each row/column, and a number: 1 for rows and 2 for columns.

Note that our models have different estimation windows, so the comparison is not completely fair because the number of `NA` is inconsistent across the columns. We need to use the option `na.rm = TRUE` :

In [63]:
```
# Means for each forecast
round(colMeans(VaR, na.rm = TRUE),3)

# Standard deviations - We
round(apply(VaR, 2, sd, na.rm = TRUE))
```

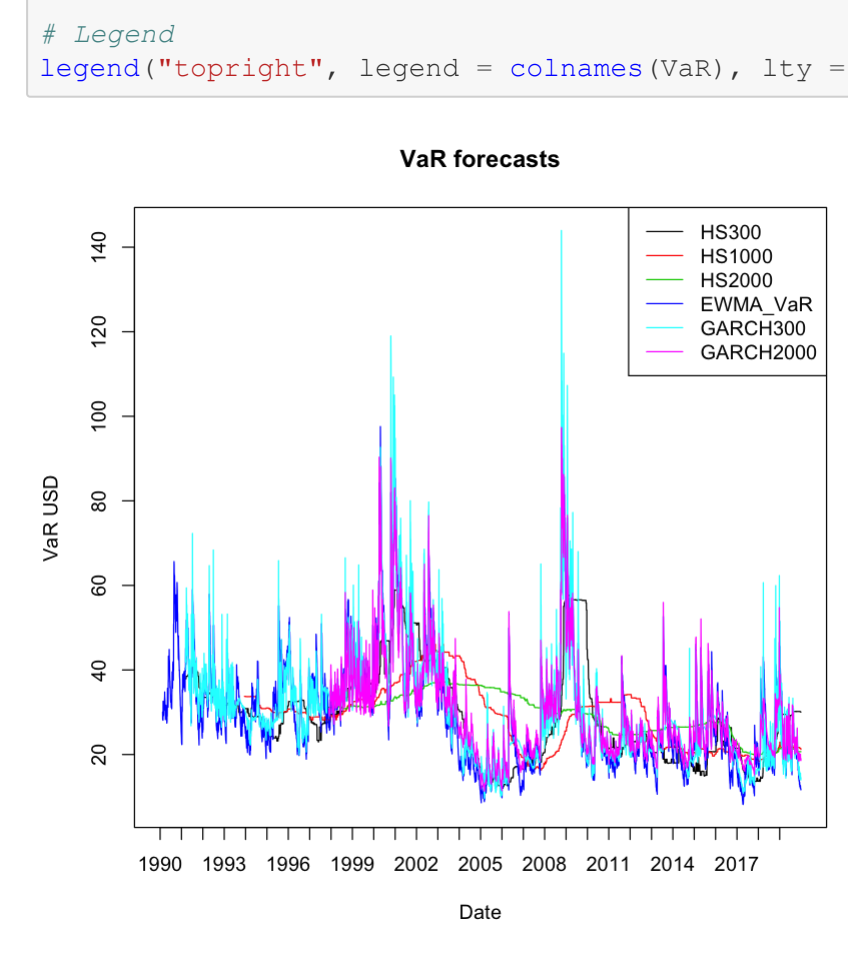
HS300	28.88
HS1000	28.844
HS2000	29.399
EWMA_VaR	29.988
GARCH300	30.193
GARCH2000	29.62

HS300	11
HS1000	8
HS2000	5
EWMA_VaR	13
GARCH300	13
GARCH2000	12

First, we see the mean is quite similar for all models. However, the standard deviations are quite different. We see that for the Historical Simulation models, a larger estimation window implies a smaller standard deviation of the forecasts. For the GARCH model, the effect of the size of the estimation window on the standard deviation of forecasts is not clear.

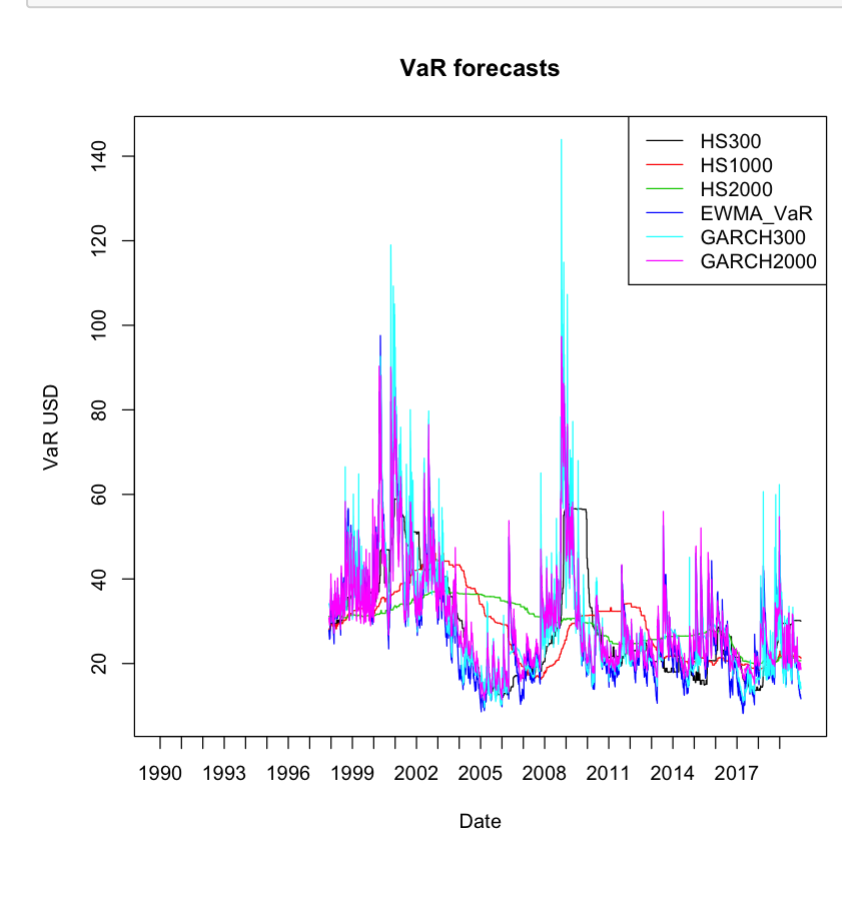
Let's use `matplot()` to plot the six forecasts:

In [64]:
```
# Plot all
matplot(dates, VaR, type = "l", lty = 1, col = 1:6, xaxt = "n", main = "VaR forecasts", xlab = "Date",
ylab = "VaR USD")
axis.Date(1, at = seq(min(dates), max(dates), by = "years"))

# Legend
legend("topright", legend = colnames(VaR), lty = 1, col = 1:6)
```



Since all our models have different estimation windows, we focus on the most restrictive one:

In [65]:
```
# Find maximum estimation window
windows <- colSums(is.na(VaR))

# Restrict to largest estimation window
VaR[1:max(windows),] <- NA

# Plot all
matplot(dates, VaR, type = "l", lty = 1, col = 1:6, xaxt = "n", main = "VaR forecasts", xlab = "Date",
ylab = "VaR USD")
axis.Date(1, at = seq(min(dates), max(dates), by = "years"))

# Legend
legend("topright", legend = colnames(VaR), lty = 1, col = 1:6)
```



Backtesting

Violation Ratios

In order to check the accuracy of our different models, we will perform a backtest. This involves comparing the forecasted VaR for time t to the realized return for that same date. If the return in time t is a loss larger than the forecasted VaR, we count this as a violation:

VaR violation: an event such that

$$\eta_t = \begin{cases} 1 & \text{if } y_t \le -\text{VaR}_t \\ 0 & \text{if } y_t > -\text{VaR}_t \end{cases}$$

We proceed to count the violations in the Testing Window W_T, which is the length of the dataset minus the Estimation Window:

Violations: v_1

Non-violations: v_0

Where:

$$v_1 = \sum_{t=1}^{W_T} \eta_t$$

$$v_0 = W_T - v_1$$

The main tool we use for backtesting are the **violation ratios**. We compare the **observed** number of VaR violations with the **expected** number. If we are calculating a 5% VaR, we expect to see $5\% \times W_T$ violations.

We define the **Violation Ratio** as:

$$\text{VR} = \frac{\text{Observed violations}}{\text{Expected violations}} = \frac{v_1}{p \times W_T}$$

If the violation ratio is greater than one, it means we observe more violations than we expect, so our VaR model is **underforecasting** risk. On the other hand, if it is smaller than one, we are **overforecasting** risk.

We expect VR = 1, but how can we interpret the VR from a model? A useful rule of thumb is:

- If VR ∈ [0.8, 1.2] the model is **good**
- If VR ∈ [0.5, 0.8] or VR ∈ [1.2, 1.5] the model is **acceptable**
- If VR ∈ [0.3, 0.5] or VR ∈ [1.5, 2] the model is **bad**
- If VR < 0.5 or VR > 2 the model is **useless**

Backtesting in R

We will now calculate the Violation Ratios for each of our models:

In [66]:
```
# Backtesting and Violation Ratios

# Let's transform VaR to a data.frame
VaR <- as.data.frame(VaR)

# Initialize a Violations data.frame, same dim and colnames as VaR, fill with NA
Violations <- VaR
Violations[] <- NA

dim(Violations)
```

7559 6

We are going to populate the Violations matrix using a loop that compares each day's VaR forecast for every model with the realized returns on that day. This is done with a `logical` , which is a type of data in R that returns `TRUE` or `FALSE` depending on the statement. Numerically, `TRUE` is equivalent to 1, and `FALSE` to 0. For example:

In [67]:
```
# Logicals in R
a <- 10000 < 5
b <- 10000 > 5

a
b

sum(a,b)
```

FALSE

TRUE

1

So we will compare each value in VaR with the realized return:

In [68]:
```
# Populating the Violations matrix

# For every model (columns in VaR)
for(i in 1:dim(VaR)[2]){

    # Fill the column in Violations with TRUE/FALSE
    # TRUE if the realized return is lower than VaR
    # FALSE otherwise
    Violations[,i] <- y*portfolio_value < -VaR[,i]
}
```

We can use the `which()` function to see where the Violations happened in every model:

In [69]:
```
# Find where violations happened
dates[which(Violations$EWMA_VaR)]
```

1997-12-18	1997-12-24	1998-03-09	1998-04-06	1998-04-23	1998-05-07	1998-05-18	1998-07-21	1998-07-28
1998-08-28	1998-08-31	1998-10-01	1998-11-30	1999-02-04	1999-03-23	1999-04-14	1999-04-19	1999-07-20
1999-09-23	2000-01-19	2000-02-09	2000-02-11	2000-03-06	2000-03-27	2000-04-03	2000-04-24	2000-05-25
2000-07-19	2000-09-14	2000-09-25	2000-09-27	2000-10-03	2000-10-16	2000-11-28	2000-11-30	2000-12-15
2001-03-12	2001-07-02	2001-07-06	2001-07-20	2001-08-17	2001-08-30	2001-09-17	2001-09-20	2001-10-09
2001-12-20	2002-01-18	2002-02-21	2002-03-20	2002-04-02	2002-04-09	2002-04-26	2002-07-22	2002-07-23
2002-07-25	2002-09-23	2002-12-09	2003-01-17	2003-01-24	2003-03-24	2003-04-09	2003-05-19	2003-06-05
2003-07-17	2003-07-21	2003-09-10	2003-09-22	2003-09-24	2003-09-30	2003-10-24	2004-01-09	2004-01-27
2004-01-28	2004-03-08	2004-07-06	2004-07-08	2004-07-23	2004-09-03	2004-10-22	2005-01-13	2005-01-19
2005-02-22	2005-03-11	2005-03-16	2005-03-29	2005-04-15	2005-04-28	2005-07-22	2005-08-08	2005-09-21
2005-10-04	2005-12-13	2006-01-20	2006-03-22	2006-04-28	2007-01-25	2007-02-05	2007-02-16	2007-02-27
2007-03-13	2007-06-07	2007-06-22	2007-07-10	2007-07-26	2007-08-09	2007-10-19	2007-11-26	2008-01-04
2008-01-08	2008-01-22	2008-02-01	2008-04-11	2008-04-25	2008-06-06	2008-07-02	2008-07-18	2008-09-16
2008-09-17	2008-09-29	2008-10-06	2008-10-07	2008-12-01	2009-01-07	2009-01-20	2009-01-22	2009-03-05
2009-06-22	2009-07-24	2009-09-01	2009-10-01	2010-01-21	2010-01-22	2010-01-29	2010-02-04	2010-03-31
2010-04-30	2010-05-04	2010-05-06	2010-05-07	2010-05-20	2010-05-26	2010-06-04	2010-06-29	2010-10-19
2010-11-23	2011-01-28	2011-03-10	2011-03-16	2011-04-29	2011-05-16	2011-06-01	2011-06-15	2011-07-27
2011-08-04	2011-08-08	2011-08-10	2011-09-21	2011-09-22	2011-11-09	2012-04-04	2012-04-10	2012-05-04
2012-05-23	2012-06-01	2012-06-11	2012-06-21	2012-06-25	2012-07-23	2012-10-09	2012-10-19	2012-10-22
2012-11-07	2012-11-13	2013-04-11	2013-06-20	2013-07-19	2013-09-03	2013-10-23	2013-12-05	2014-01-06
2014-01-13	2014-02-03	2014-04-04	2014-04-10	2014-09-25	2014-10-09	2014-10-10	2014-12-16	2015-01-27
2015-03-25	2015-06-29	2015-07-22	2015-08-06	2015-08-20	2015-08-21	2015-08-24	2015-09-01	2015-12-18
2016-01-07	2016-01-15	2016-02-05	2016-04-22	2016-06-13	2016-06-24	2016-06-27	2016-09-09	2016-10-11
2016-11-10	2017-02-01	2017-03-02	2017-03-21	2017-05-11	2017-05-17	2017-06-09	2017-06-27	2017-06-29
2017-08-10	2017-08-17	2017-09-25	2017-11-29	2017-12-04	2018-02-02	2018-02-05	2018-02-08	2018-03-22
2018-03-23	2018-03-27	2018-06-25	2018-07-30	2018-09-05	2018-10-04	2018-10-10	2018-10-24	2018-12-07
2018-12-24	2019-03-22	2019-05-01	2019-05-07	2019-05-13	2019-06-03	2019-06-25	2019-07-31	2019-08-05
2019-08-14	2019-08-23							

Let's pick one of the days where HS2000 is violated and see if the other models also violate VaR:

In [70]:
```
# Get a random day where EWMA VaR is violated using sample()
random_day <- sample(dates[which(Violations$HS2000)],1)

# Find the index in dates using which()
day_index <- which(dates == random_day)

# See that row in Violations
paste0("Violation for HS2000 on ",random_day)
Violations[day_index,]
```

'Violation for HS2000 on 2011-08-10'

	HS300	HS1000	HS2000	EWMA_VaR	GARCH300	GARCH2000
5448	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

We can easily plot the VaR forecasts and add points where the Violations happened using the `points()` function.

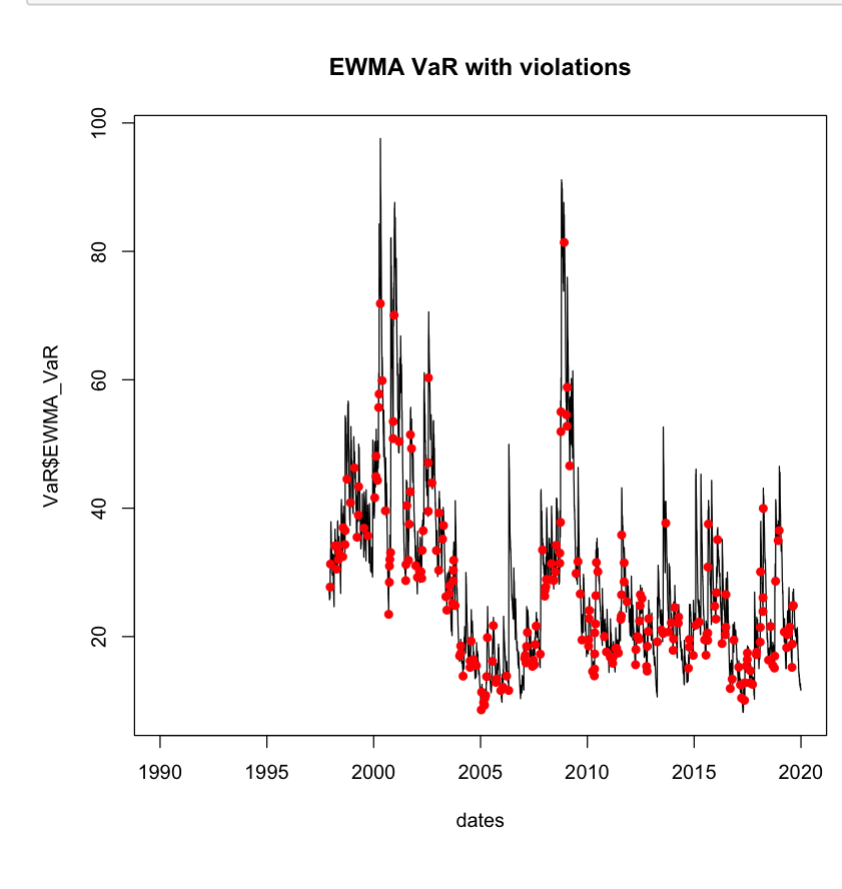
A useful feature about logical vectors is that we can use them to subset other objects. For example:

In [71]:
```
# Subsetting with logical vectors
logi <- c(TRUE, FALSE, TRUE, TRUE)
vec <- c(1,2,3,4)

# If we subset by "logi", we will only keep positions where TRUE
vec[logi]
```

1 3 4

We can use this feature to subset our objects to where violations happened:

In [72]:
```
# Plotting the violations
plot(dates, VaR$EWMA_VaR, type = "l", main = "EWMA VaR with violations")

# Add points where the violations happened
points(dates[Violations$EWMA_VaR], VaR$EWMA_VaR[Violations$EWMA_VaR], pch = 16, col = "red")
```



Using the `&` operator, let's see if they are dates for which all our models have a VaR violation. We can use `apply()` and the `all()` function, which checks if all elements are `TRUE` :

In [73]:
```
# Check dates where all models have a violation
w <- apply(Violations, 1, all)
```

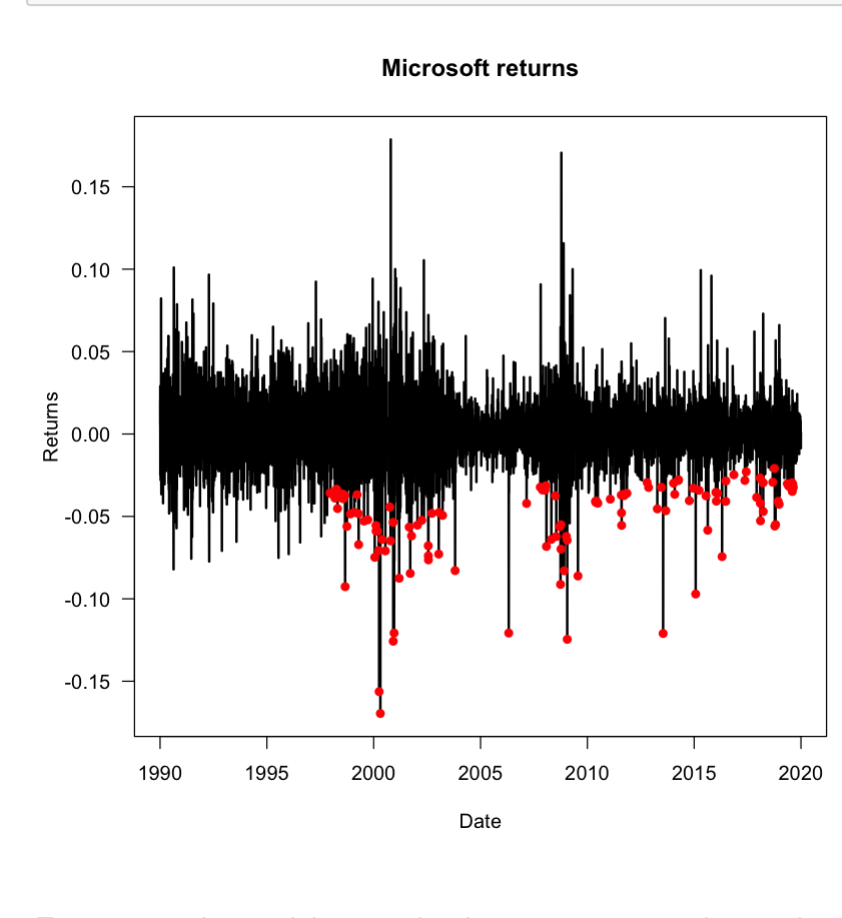
We can use `sum()` to count the number of days where all models have a violation:

In [74]:
```
# Days where all models have a violation
sum(w, na.rm = TRUE)
```

122

Let's plot these days with the returns:

In [75]:
```
# Plotting the returns and adding the days where all models had a violation
plot(dates, y, main = "Microsoft returns", type = "l", lwd = 2, las = 1,
     xlab = "Date", ylab = "Returns")
points(dates[w], y[w], pch = 16, col = "red")
```



To compare the models to each other, we can count the number of violations for every model using `colSums()` :

In [76]:
```
# Counting Violations by model
colSums(Violations, na.rm = TRUE)
```

HS300	295
HS1000	300
HS2000	264
EWMA_VaR	245
GARCH300	232
GARCH2000	210

Now we can create a VR object that holds the Violation Ratio for every model:

In [77]:
```
# Creating a Violation Ratio object

# Remove the rows with NA
Violations <- Violations[!is.na(Violations[,1]),]

# Get the column sums
V <- colSums(Violations)

# Calculate expected violations
EV <- dim(Violations)[1]*p

# Violation Ratios
VR <- V/EV

# Call object, rounding to 3 decimals
round(VR,3)
```

HS300	1.061
HS1000	1.079
HS2000	0.95
EWMA_VaR	0.881
GARCH300	0.835
GARCH2000	0.756

In [78]:
```
# We can write a function that uses our rule of thumb to assess the model
model_assessment <- function(VR) {
    if (VR > 0.8 & VR < 1.2) {
        paste0(names(VR), "Model is good")
    } else if ((VR > 0.5 & VR <= 0.8) | (VR > 1.2 & VR <= 1.5)) {
        paste0(names(VR), "Model is acceptable")
    } else if ((VR > 0.3 & VR <= 0.5) | (VR > 1.5 & VR <= 2)) {
        paste0(names(VR), "Model is bad")
    } else {
        paste0(names(VR), "Model is useless")
    }
}
```

In [79]:
```
# We can use sapply(), the vector version of apply()
sapply(VR, model_assessment)
```

HS300	'Model is good'
HS1000	'Model is good'
HS2000	'Model is good'
EWMA_VaR	'Model is good'
GARCH300	'Model is good'
GARCH2000	'Model is acceptable'

All our models fall under the *good* category except for the GARCH with 2000 days as estimation window, which is only *acceptable*. The best performing model under this criteria is:

In [80]:
```
# Best performing - VR closest to 1
sort(round(abs(VR-1),3))
```

HS2000	0.05
HS300	0.061
HS1000	0.079
EWMA_VaR	0.119
GARCH300	0.165
GARCH2000	0.244

The model with the best Violation Ratio is the Historical Simulation with 2000 days for Estimation Window. This is interesting since it was the model with the smallest standard deviation.

Multivariate EWMA and HS VaR

We will now perform a Multivariate VaR forecast using EWMA and HS for Microsoft, JP Morgan, and Intel, using an estimation window of $W_E = 1000$:

In [81]:
```
# Multivariate EWMA and HS VaR

# Determine a vector of portfolio weights
w <- c(0.5, 0.2, 0.3)

# EWMA VaR
multi_y <- Y[,c("MSFT", "JPM", "INTC")]
multi_y <- as.matrix(multi_y)
n <- dim(multi_y)[1]
K <- dim(multi_y)[2]

# Number of variables
nvar <- K+K*(K-1)/2
EWMA <- matrix(NA, nrow = n, ncol = nvar)
lambda <- 0.94
S <-- cov(multi_y)
# Fill initial row, order:
# 1st col: MSFT Variance, 2nd: JPM Variance, 3rd: INTC Variance
# 4th: MSFT-JPM, 5th: MSFT-INTC, 6th: JPM-INTC

EWMA[1,] <- c(S[1,1], S[2,2], S[3,3], S[1,2], S[1,3], S[2,3])
colnames(EWMA) <- c("MSFT", "JPM", "INTC", "MSFT-JPM", "MSFT-INTC", "JPM-INTC")
head(EWMA)
```

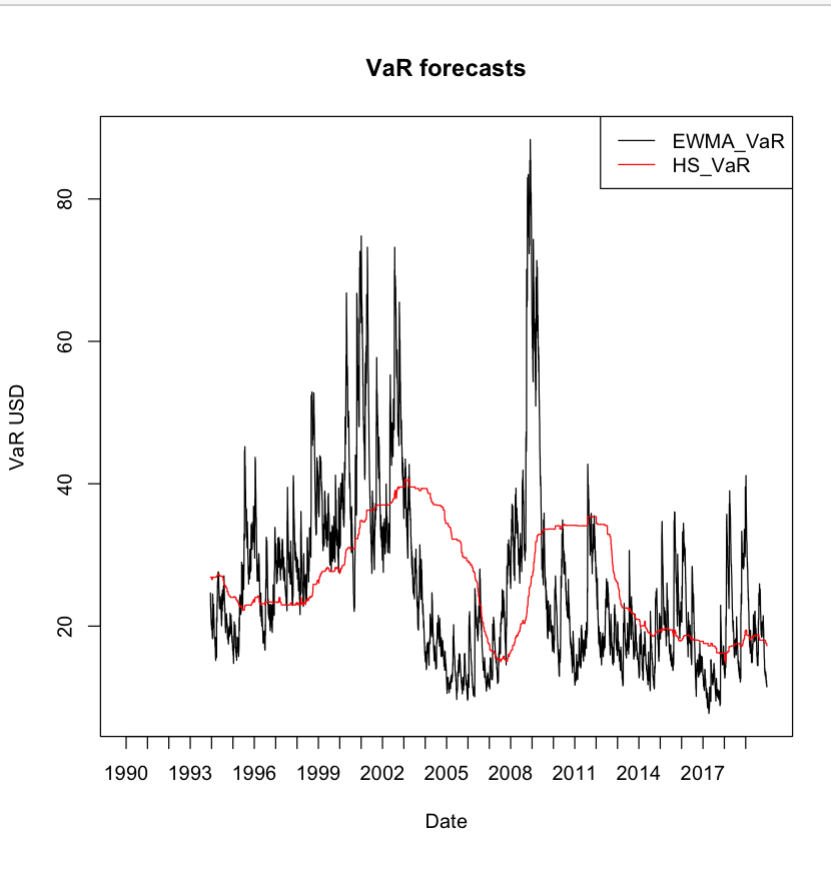
MSFT	JPM	INTC	MSFT-JPM	MSFT-INTC	JPM-INTC
-0.0003950531	-0.0005487714	-0.0005604875	-0.0001838818	-0.0002649186	-0.000214687
NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA

In [83]:
```
# Populating EWMA matrix and creating the portfolio variance

portfolio_var <- rep(NA,n)

for (i in 2:n) {
    S <- lambda * S + (1-lambda) * multi_y[i-1,] %*% t(multi_y[i-1,])
    EWMA[i,] <- c(S[1,1], S[2,2], S[3,3], S[1,2], S[1,3], S[2,3])
    portfolio_var[i] <- t(w) %*% S %*% w
}

# Implement the VaR forecast
# Plot estimation for conditional volatility
EWMA_cond_volatility <- sqrt(portfolio_var)
EWMA_VaR <- -qnorm(p) * EWMA_cond_volatility * portfolio_value

# Burn the first 1000 observations
EWMA_VaR[1:1000] <- NA
```

In [84]:
```
# Multivariate HS

# Initialize the vector
HS_VaR <- rep(NA, length = n)

# Do the HS
window <- 1000
for (i in 1:(n-window)) {
    # Multiply the matrix y (dimension T x 3) by vector w (dimension 3x1)
    # Output is portfolio returns (dimension Tx1)
    yp <- multi_y[i:(i+window),] %*% w
    head(yp)

    # Sort the vector
    ys <- sort(yp)

    # Position of the 5% quantile
    quant <- ceiling(p*length(ys))

    # Get VaR
    HS_VaR[i+window] <- -ys[quant]*portfolio_value
}
```

Now that we have implemented both EWMA_VaR and HS_VaR, we can plot them together, find the mean and SD, and evaluate the Violation Ratios:

In [85]:
```r
multi_VaR <- cbind(EWMA_VaR, HS_VaR)

matplot(dates, multi_VaR, type = "l", lty = 1, col = 1:2, xaxt = "n", main = "VaR forecasts", xlab = "Date", ylab = "VaR USD")
axis.Date(1, at = seq(min(dates), max(dates), by = "years"))

# Legend
legend("topright", legend = colnames(multi_VaR), lty = 1, col = 1:2)
```



In [86]:
```r
# Mean and SD
round(colMeans(multi_VaR, na.rm = TRUE))
round(apply(multi_VaR, 2, sd, na.rm = TRUE))
```

EWMA_VaR	26
HS_VaR	26

EWMA_VaR	13
HS_VaR	7

As expected, the means are very close to each other but the HS forecast has a lower standard deviation.

In [87]:
```r
# Violation Ratios

# Transform VaR to data frame
multi_VaR <- as.data.frame(multi_VaR)

multi_Violations <- multi_VaR

multi_Violations[] <- NA

for(i in 1:dim(multi_VaR)[2]){
    returns <- multi_y %*% w
    multi_Violations[,i] <- returns*portfolio_value < -multi_VaR[,i]
}

multi_Violations <- multi_Violations[!is.na(multi_Violations[,1]),]

multi_V <- colSums(multi_Violations)

multi_EV <- dim(multi_Violations)[1]*p

multi_VR <- multi_V/multi_EV

round(multi_VR,3)

model_assessment(multi_VR[1])
model_assessment(multi_VR[2])
```

EWMA_VaR	0.994
HS_VaR	1.061

'EWMA_VaRModel is good'

'HS_VaRModel is good'

Both models are good, with Violation Ratios close to 1.

Stress Testing

We are interested in seeing how our models perform in stressful periods of time. So we will define a new time period that consists of the years of the financial crisis (2008-2012), and evaluate our four univariate models then:

In [90]:
```r
# Stress Testing

# Subset for crisis peridos
crisis <- year(dates) >= 2008 & year(dates) < 2013
y_crisis <- y[crisis]
VaR_crisis <- VaR[crisis,]

Violations_crisis <- VaR_crisis
Violations_crisis[] <- NA

for(i in 1:dim(VaR_crisis)[2]){
    Violations_crisis[,i] <- y_crisis*portfolio_value < -VaR_crisis[,i]
}


# Remove the rows with NA
Violations_crisis <- Violations_crisis[!is.na(Violations_crisis[,1]),]

# Get the column sums
V_crisis <- colSums(Violations_crisis)

# Calculate expected violations
EV_crisis <- dim(Violations_crisis)[1]*p

# Violation Ratios
VR_crisis <- V_crisis/EV_crisis

# Call object, rounding to 3 decimals
round(VR_crisis,3)

sapply(VR_crisis, model_assessment)
```

HS300	1.144
HS1000	1.35
HS2000	1.176
EWMA_VaR	1.048
GARCH300	1.064
GARCH2000	0.937

HS300	'Model is good'
HS1000	'Model is acceptable'
HS2000	'Model is good'
EWMA_VaR	'Model is good'
GARCH300	'Model is good'
GARCH2000	'Model is good'

In this test, all models are good except for HS1000. The model with the Violation Ratio closest to 1 is the EWMA, followed by GARCH2000 which was the worst perfoming model when we considered the entire time period.

In [92]:
```r
# Best performing - VR closest to 1
sort(round(abs(VR_crisis-1),3))
```

EWMA_VaR	0.048
GARCH2000	0.063
GARCH300	0.064
HS300	0.144
HS2000	0.176
HS1000	0.35

Recap

In this seminar we have covered:

- VaR implementation with EWMA, HS and GARCH
- Comparison of different models and estimation windows
- Backtesting using Violation Ratios
- Assessing models based on their VR
- Multivariate VaR forecast and assessment using EWMA and HS
- Stress-Testing

Some new functions used:

- `DoGARCH()`
- `apply()`
- `sapply()`
- `model_assessment()`

For more discussion on the material covered in this seminar, refer to *Chapter 5: Implementing risk forecasts* and *Chapter 8: Backtesting and stress testing* on *Financial Risk Forecasting* by Jon Danielsson.

Acknowledgements: Thanks to Alvaro Aguirre for creating these notebooks