Financial Risk Forecasting
Seminar Week 10: Backtesting Risk Models

Jon Danielsson
London School of Economics

Version 4.0

10 Backtesting Risk Models

10.1 Why backtesting matters for financial risk

Backtesting is the systematic evaluation of risk model performance using historical data. After developing VaR models in Weeks 8-9, we must validate their accuracy before using them for real-world risk management decisions.

The key insight is that a good VaR model should have violations (actual losses exceeding VaR) occur at the expected frequency.

Building on Week 8's risk forecasting methods and Week 9's simulation approaches, we now test these models against actual market outcomes to evaluate their practical effectiveness.

For more detail, see Backtesting in the R notebook.

10.2 The plan for this week

1. Set up systematic backtesting framework for storing forecasts and violations
2. Implement rolling-window backtesting for HS, EWMA and GARCH VaR models
3. Calculate violation rates and compare model performance
4. Analyse results graphically and statistically

10.3 Loading data and libraries

```
load('Returns.RData')
load('Prices.RData')
```

```
library(rugarch)
library(lubridate)
```

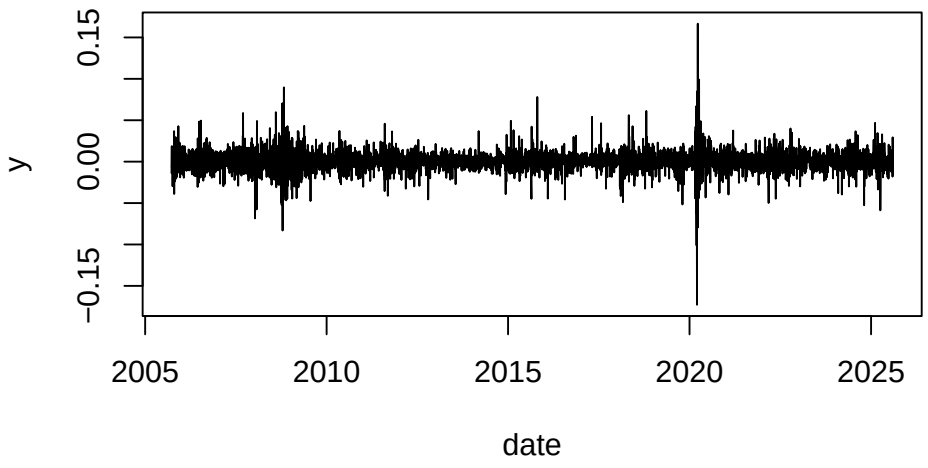
10.4 Set parameters

We need to specify the parameters for the VaR models and the backtesting procedure.

```
p=0.01          # VaR probability level (1% VaR)
lambda=0.94     # EWMA decay parameter
value=1         # Portfolio value
T=5000          # Total number of observations
WE=1000         # Estimation window size
WT=5000-WE      # Testing window size (4000 observations)
Burn=30         # Burn-in period for EWMA
```

Pick `MCD` for analysis.

```
y=tail(Returns$MCD,T)
d=tail(Returns$date,T)
date=ymd(d)
plot(date,y,type='l')
```



10.5 Keeping track of the forecasts

We store all the forecasts in a `data frame` called `VaR`. We start by putting the returns into it and then add the various methods as new columns. This ensures that all the dates match up correctly.

```
VaR=as.data.frame(matrix(NA,ncol=1,nrow=length(y)))
names(VaR)=c("y")
VaR$y=y
VaR$date=date
```

10.6 HS

Create a new column in the `VaR` data frame, fill it with `NA` and call it `HS`.

```
VaR$HS=NA
```

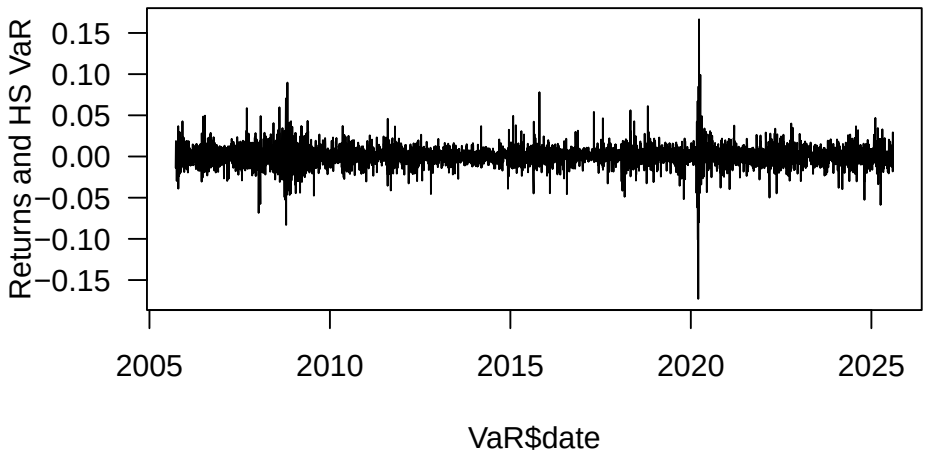
We can then run the HS forecasts. Note that it is very easy to make a mistake

with the windows and the forecast date. By using the explicit approach below, we minimise the chance of mistakes. To ensure the data is correct, it can be useful to print it in the loop.

```
for(t in (WE+1):T) {
  t1=t-WE
  t2=t-1
  window=VaR$y[t1:t2]
# to debug
#  cat(t,t1,t2,length(window),'\n')
  VaR$HS[t]=-sort(window)[WE*p]*value
}
```

We can then plot the returns with the HS forecasts superimposed.

```
plot(VaR$date,VaR$y,type='l',las=1,ylab='Returns and HS VaR')
lines(-VaR$HS,type='s',col="red",lwd=2)
```



10.7 EWMA

We set EWMA up in the same way as HS and initialise the first value to the unconditional variance.

```
VaR$EWMA=NA
VaR$EWMA[1]=var(y)
```

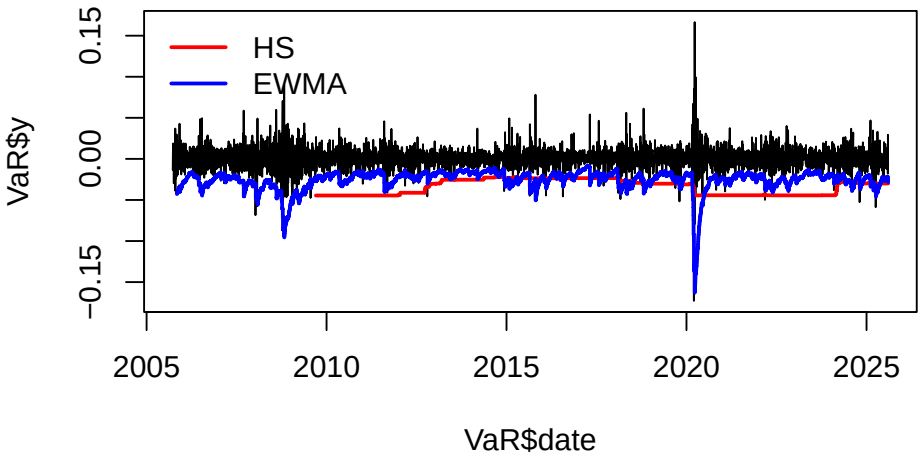
We can then run the loop.

```
for(t in 2:T){
  VaR$EWMA[t]=
    lambda*VaR$EWMA[t-1]+
    (1-lambda) *VaR$y[t-1]^2
}
VaR$EWMA=- sqrt(VaR$EWMA) * qnorm(p) * value
```

The first few observations of the forecast are not valid and we set them to `NA`. This is the burn period.

We then plot it.

```
VaR$EWMA[1:Burn]=NA
plot(VaR$date,VaR$y,type='l')
lines(VaR$date,-VaR$HS,type='s',col="red",lwd=2)
lines(VaR$date,-VaR$EWMA,type='s',col="blue",lwd=2)
legend("topleft",legend=c("HS","EWMA"),col=c("red","blue"),lty=1,bty="n",lwd=2)
```



10.8 GARCH

We proceed with the GARCH model in the same way. Note, we have to calculate the one-day-ahead forecasts of the volatility and use that for the VaR.

```
spec=ugarchspec(
  mean.model = list(
    armaOrder=c(0,0),
    include.mean=FALSE)
)
```

```
VaR$GARCH=NA
for(t in (WE+1):T) {
  t1=t-WE
  t2=t-1
  window=VaR$y[t1:t2]

  fit=ugarchfit(spec=spec,data=window,solver = "hybrid")
  s2=coef(fit)[1] +
    coef(fit)[2] * tail(window,1)^2 +
    coef(fit)[3] *tail(fit@fit$var,1 )
  VaR$GARCH[t]=-value*qnorm(p,sd=sqrt(s2))
}
```

Note which values are `NA`:

```
VaR[1,]
```

```
            y       date HS EWMA GARCH
```

```
1 0.05178566 2005-09-22 NA   NA    NA
```

```
VaR[29:32,]
```

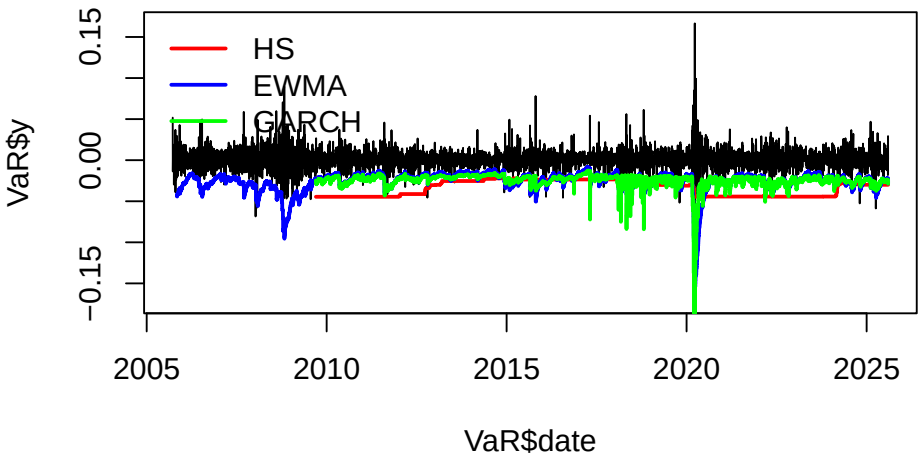
```
             y       date HS        EWMA GARCH
29 0.005369340 2005-11-01 NA          NA    NA
30 0.015303245 2005-11-02 NA          NA    NA
31 0.029626734 2005-11-03 NA 0.04216011    NA
32 0.006001392 2005-11-04 NA 0.04422490    NA
```

```
VaR[999:1002,]
```

```
                y       date          HS       EWMA      GARCH
999  -0.004365501 2009-09-10          NA 0.02416956         NA
1000 -0.008422512 2009-09-11          NA 0.02356493         NA
1001 -0.002946649 2009-09-14 0.04455073 0.02334571 0.02524845
1002  0.013735354 2009-09-15 0.04455073 0.02269670 0.02465393
```

And plot returns and all the VaRs.

```
plot(VaR$date,VaR$y,type='l')
lines(VaR$date,-VaR$HS,type='s',col="red",lwd=2)
lines(VaR$date,-VaR$EWMA,type='s',col="blue",lwd=2)
lines(VaR$date,-VaR$GARCH,type='s',col="green",lwd=2)
legend("topleft",
    legend=c("HS","EWMA","GARCH"),
    col=c("red","blue","green"),
    lty=1,
    bty="n",lwd=2)
```



10.9 Violations Analysis

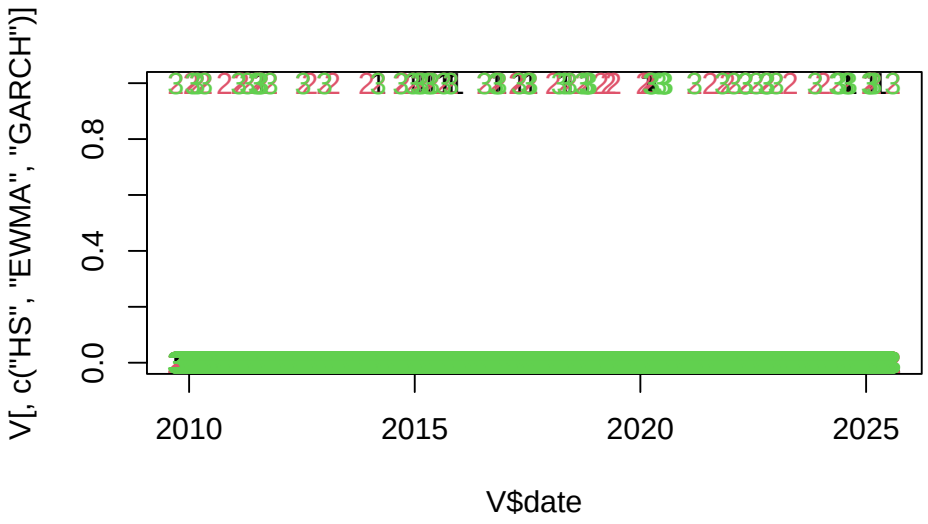
We create a systematic framework for comparing violations across all three methods, starting with making a separate data frame for the violations, called `V`, by copying `VaR` over the testing window.

```
V=VaR[(WE+1):T,]
for(i in c("HS","EWMA","GARCH")){
```

```
  V[,i]=V[,"y"]-V[,i]
  V[V[,i]<0,i]=0
  V[V[,i]>0,i]=1
}
```

We can then plot the violations and print the violation ratios.

```
matplot(V$date,V[,c("HS","EWMA","GARCH")])
```




```
colSums(V[,c("HS","EWMA","GARCH")])
```

```
   HS  EWMA GARCH
   33    69    58
```

```
colSums(V[,c("HS","EWMA","GARCH")])/WT/p
```

```
   HS  EWMA GARCH
0.825 1.725 1.450
```

10.10 Statistical Testing: The Bernoulli Coverage Test

While violation ratios provide initial insight into model performance, we need formal statistical tests to determine whether deviations from the expected violation rate are statistically significant or due to random variation.

10.10.1 The coverage testing framework

The Bernoulli coverage test, developed by Christoffersen (1998), tests whether violations occur at the expected frequency. Under the null hypothesis, violations should follow a Bernoulli distribution with probability equal to the VaR level (1% in our case).

The test uses a likelihood ratio approach:

- Under the null hypothesis: violations occur with probability p (our chosen VaR level)

- Under the alternative: violations occur with probability $\hat{p}$ (the observed frequency)

```
bern_test=function(p,v){
  lv=length(v)      # Length of violations vector
  sv=sum(v)         # Number of violations
  # Log-likelihood under the alternative (using observed frequency)
  al=log(p)*sv+log(1-p)*(lv-sv)
  # Log-likelihood under the null (using expected frequency)
  bl=log(sv/lv)*sv +log(1-sv/lv)*(lv-sv)
  # Likelihood ratio test statistic
  return(-2*(al-bl))
}
```

10.10.2 Applying the coverage test

We now apply the Bernoulli coverage test to each of our three VaR models. The test statistic follows a chi-squared distribution with 1 degree of freedom. At the 5% significance level, we reject the null hypothesis if the test statistic exceeds 3.84.

```
# Critical value at 5% significance level
critical_value = qchisq(0.95, 1)
cat("Critical value (5% level):", critical_value, "\n\n")
```

```
Critical value (5% level): 3.841459
```

```
# Apply test to each method
methods = c("HS", "EWMA", "GARCH")
for(method in methods) {
  test_stat = bern_test(p, V[,method])
  violation_rate = sum(V[,method])/length(V[,method])

  cat(method, ":\n")
  cat("  Violation rate:", round(violation_rate*100, 2), "%\n")
  cat("  Expected rate:", p*100, "%\n")
  cat("  Test statistic:", round(test_stat, 3), "\n")
  cat("  p-value:", round(1-pchisq(test_stat, 1), 4), "\n")

  if(test_stat > critical_value) {
    cat("  Result: REJECT null hypothesis - model inadequate\n")
  } else {
    cat("  Result: Cannot reject null hypothesis - model acceptable\n")
  }
  cat("\n")
}
```

```
HS :
  Violation rate: 0.83 %
  Expected rate: 1 %
  Test statistic: 1.316
  p-value: 0.2513
```

```
  Result: Cannot reject null hypothesis - model acceptable

EWMA :
  Violation rate: 1.73 %
  Expected rate: 1 %
  Test statistic: 17.454
  p-value: 0
  Result: REJECT null hypothesis - model inadequate

GARCH :
  Violation rate: 1.45 %
  Expected rate: 1 %
  Test statistic: 7.183
  p-value: 0.0074
  Result: REJECT null hypothesis - model inadequate
```

10.10.3 Interpreting the results

The Bernoulli coverage test helps us distinguish between:

- Statistical noise: Small deviations from 1% that are within expected random variation
- Model misspecification: Systematic over- or under-estimation of risk that requires model adjustment

A model that fails the coverage test may be:

- Too conservative (too few violations): Overestimating risk, leading to excessive capital requirements
- Too aggressive (too many violations): Underestimating risk, creating dangerous exposure

The test provides objective criteria for model selection and regulatory compliance, as Basel requirements specify acceptable violation frequencies for internal models.

10.11 Recap

10.11.1 In this seminar, we have covered:

- Understanding backtesting as essential validation for the VaR models developed in Weeks 8-9
- Systematic backtesting framework design:
  - Creating data frames to store forecasts and track violations consistently
  - Proper windowing for estimation and testing periods
  - Ensuring fair comparison across different models
- Implementing rolling-window backtesting for three risk models:
  - Historical Simulation (HS) with rolling quantile estimation
  - EWMA with recursive volatility updates and burn-in handling
  - GARCH with one-step-ahead conditional volatility forecasts
- Violation analysis methodology:

  - Converting forecast errors to binary violation indicators
  - Calculating violation ratios as key performance metrics
  - Systematic comparison across HS, EWMA and GARCH methods
- Statistical testing with the Bernoulli coverage test:
  - Implementing likelihood ratio tests for violation frequencies
  - Distinguishing statistical noise from model misspecification
  - Applying chi-squared distribution for hypothesis testing
  - Interpreting test results for regulatory compliance
- Model validation principles:
  - Understanding that good VaR models should have violations at expected frequency
  - Recognizing when models are too conservative or inadequate
  - Using formal statistical tests for objective model evaluation
  - Meeting Basel framework requirements for internal models
- Graphical evaluation techniques:
  - Time series plots showing returns with VaR estimates
  - Violation pattern analysis across different market conditions
  - Comparative visualization of model performance

10.11.2 Some new functions used:

- `colSums()` — calculate column sums of a matrix
- `qchisq()` — quantile function for chi-squared distribution

10.12 Optional exercises

1. Multi-asset backtesting comparison:
   - Repeat the analysis for all stocks in `Returns.RData`
   - Create a summary table showing violation ratios for each stock and method
   - Which stocks are hardest to forecast? Does this vary by method?
   - Create a heatmap showing violation ratios across stocks and methods
2. Statistical testing of violations:
   - Extend the Bernoulli coverage test to handle multiple significance levels
   - Implement the Christoffersen test for independence of violations
   - Apply both tests to all three methods (HS, EWMA, GARCH)
   - At what significance level can you reject the correct specification?
3. Comprehensive VaR backtesting function:
   - Create `backtest_VaR(returns, method, WE, p, value)` that:
   - Accepts method = "HS", "EWMA" or "GARCH"
   - Returns a list with VaR forecasts, violations and test statistics
   - Include options for different GARCH specifications
   - Add visualisation of results as part of the output
4. Window size sensitivity analysis:
   - Create a function to test WE = c(250, 500, 750, 1000, 1250, 1500)
   - For each window size, calculate the violation ratio and the coverage test p-value
   - Plot violation ratio vs window size for each method
   - What is the optimal window size for each method?

5. Model comparison and visualization:
    - Create a function that saves comparison plots in multiple formats
    - Include: VaR time series plot, violation scatter plot, QQ plots of violations
    - Save as PNG (for presentations), SVG (for web), PDF (for papers)
    - Add annotations showing violation ratios and test statistics on plots
6. Expected Shortfall backtesting:
    - Extend the analysis to include ES backtesting
    - Implement the Acerbi & Szekely ES backtesting approach
    - Compare ES violation magnitudes across methods
    - Which method best captures tail risk beyond VaR?
7. Dynamic model selection:
    - Implement a backtesting procedure that switches between models
    - Use a 100-day rolling window to evaluate recent performance
    - Switch to the best-performing model based on the violation ratio
    - Does dynamic selection outperform any single model?
8. Professional reporting with Quarto:
    - Create a complete backtesting report template
    - Include executive summary, methodology, results and conclusions
    - Generate both Word and PDF versions with consistent formatting
    - Add a presentation version (PowerPoint and Beamer) with key findings
    - Include automated interpretation of test results