Stochastic Processes and Long-Run Behavior

This investigation studies long-run behavior in stock series using volatility clustering, ARCH(2) simulation, and regime-based forecasting.

```r
set.seed(1234)
data("EuStockMarkets")
```

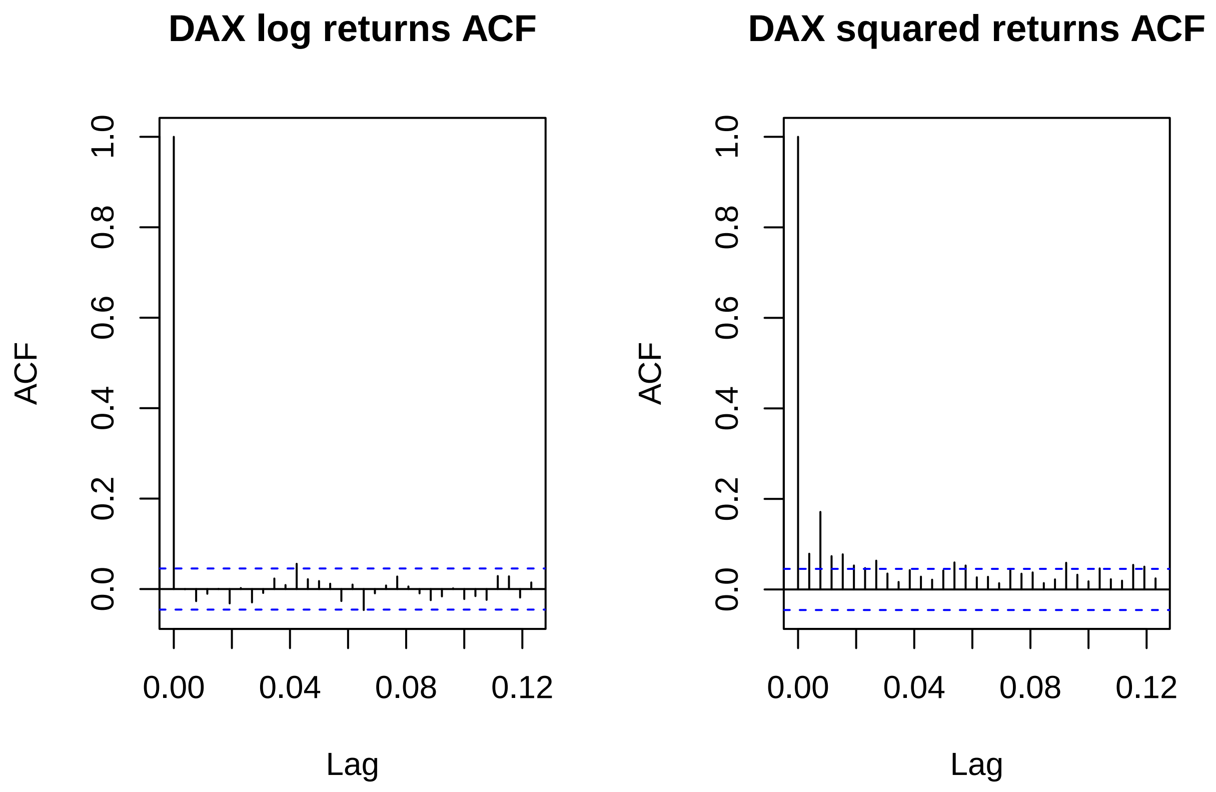
Volatility clustering and ARCH(2) simulation

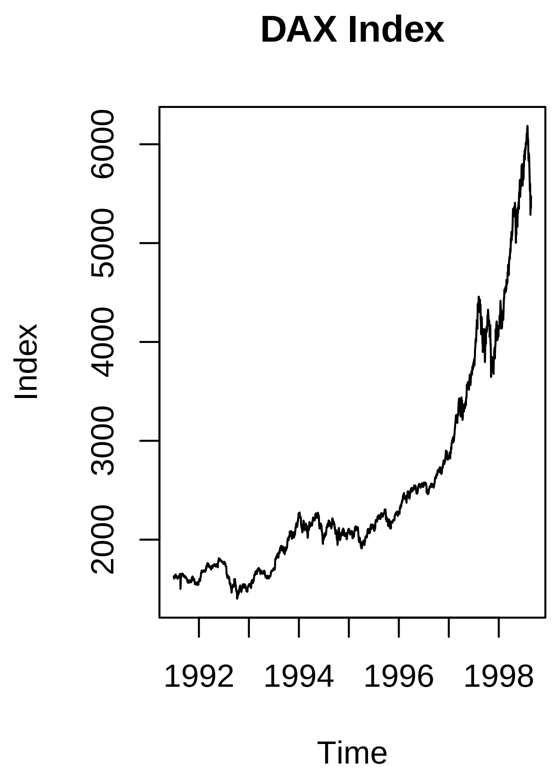
We analyze each index by: 1) computing log returns and drift, 2) checking ACF for returns and squared returns, 3) fitting an ARCH(2) proxy using AR(2) on squared returns, and 4) simulating a synthetic index with the same length and drift.

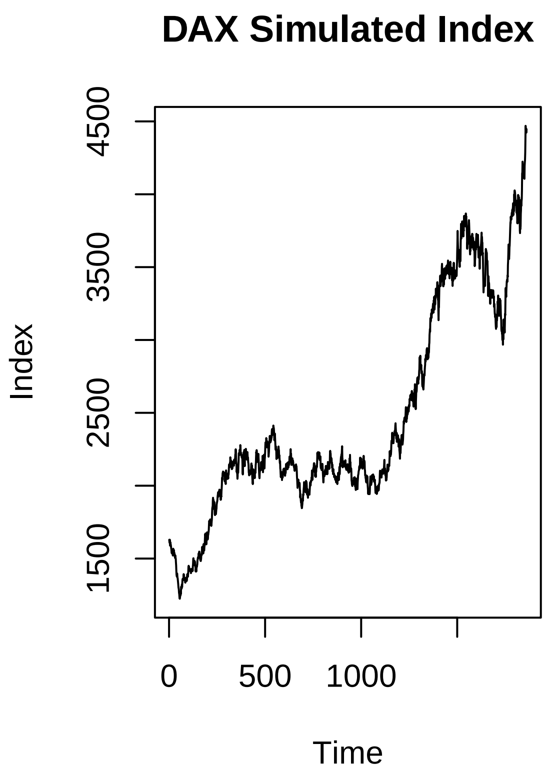
```r
analyze_arch2 <- function(index_series, label) {
  log_returns <- diff(log(index_series))
  drift <- mean(log_returns)

  par(mfrow = c(1, 2))
  acf(log_returns, main = paste(label, "log returns ACF"))
  acf(log_returns^2, main = paste(label, "squared returns ACF"))

  fit <- arima(log_returns^2, order = c(2, 0, 0))
  phi <- fit$coef[1:2]
  a0 <- fit$coef[3]

  n <- length(index_series)
  y <- numeric(n)
  y[1:2] <- log_returns[1:2]
  z <- rnorm(n)

  for (i in 3:n) {
    s <- sqrt(a0 + phi[1] * y[i - 1]^2 + phi[2] * y[i - 2]^2)
    y[i] <- s * z[i]
  }

  y <- y + drift
  y <- c(log(index_series[1]), y)
  sim_index <- exp(cumsum(y))

  par(mfrow = c(1, 2))
  ts.plot(index_series, main = paste(label, "Index"), ylab = "Index")
  ts.plot(sim_index, main = paste(label, "Simulated Index"), ylab = "Index")

  invisible(list(
    log_returns = log_returns,
    drift = drift,
    fit = fit,
    sim_index = sim_index
  ))
}
```

DAX

```
dax <- EuStockMarkets[, "DAX"]
dax_out <- analyze_arch2(dax, "DAX")
```



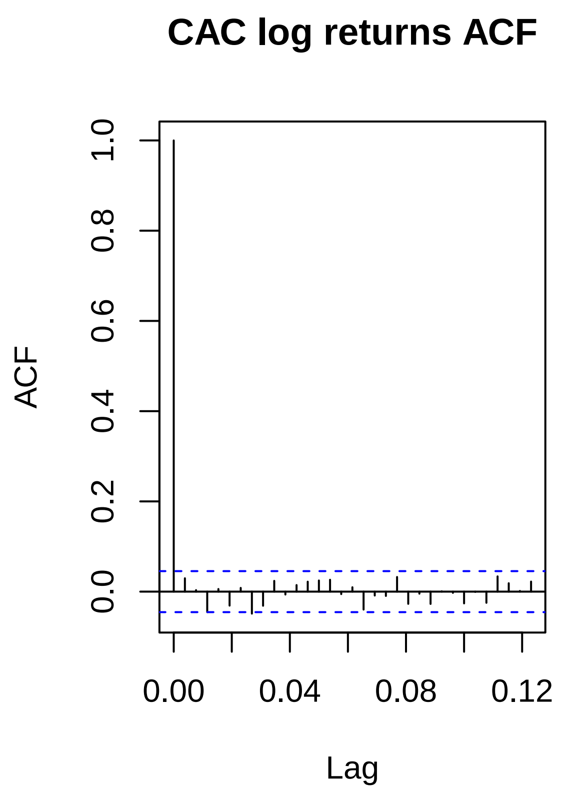




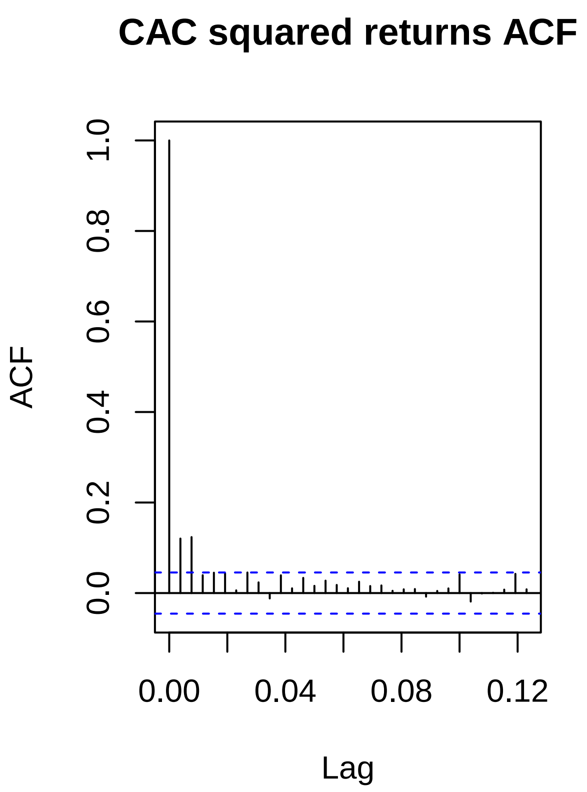
The DAX log-return ACF does not show meaningful spikes beyond lag 0, so there is no evidence of linear dependence. The squared returns ACF shows significant early-lag autocorrelation, indicating volatility clustering.

CAC

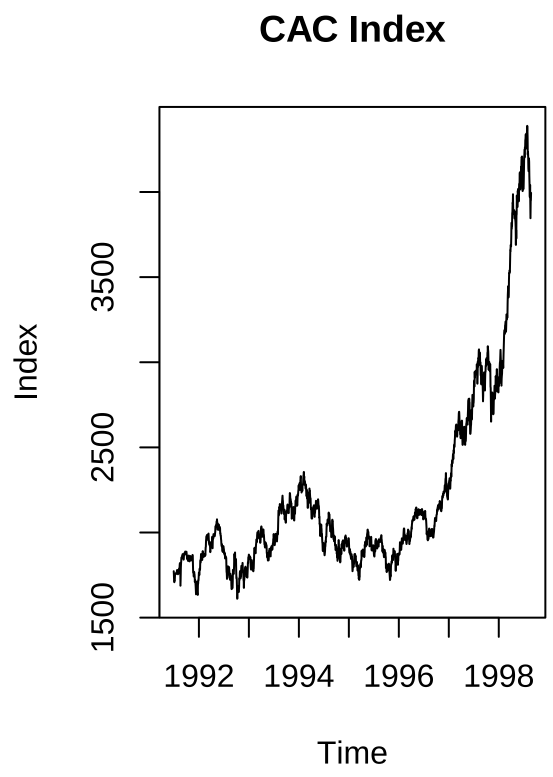
```
cac <- EuStockMarkets[, "CAC"]
cac_out <- analyze_arch2(cac, "CAC")
```

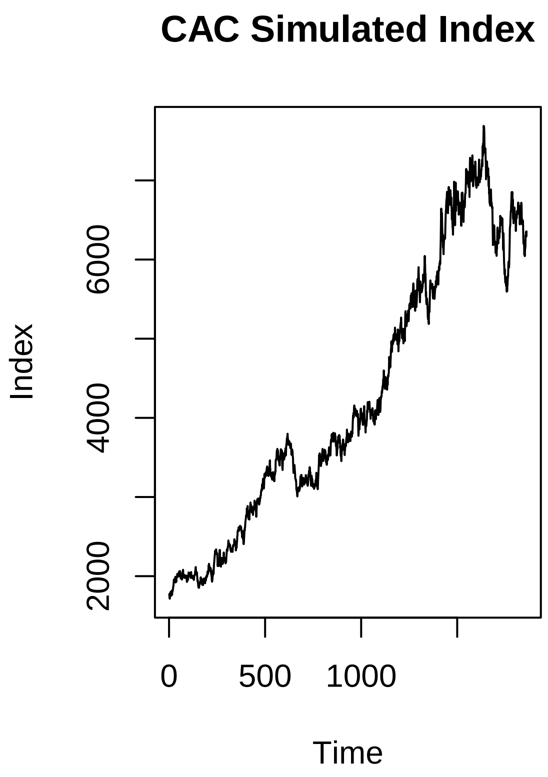


CAC log returns ACF
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.04
0.08
0.12



CAC squared returns ACF
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.04
0.08
0.12

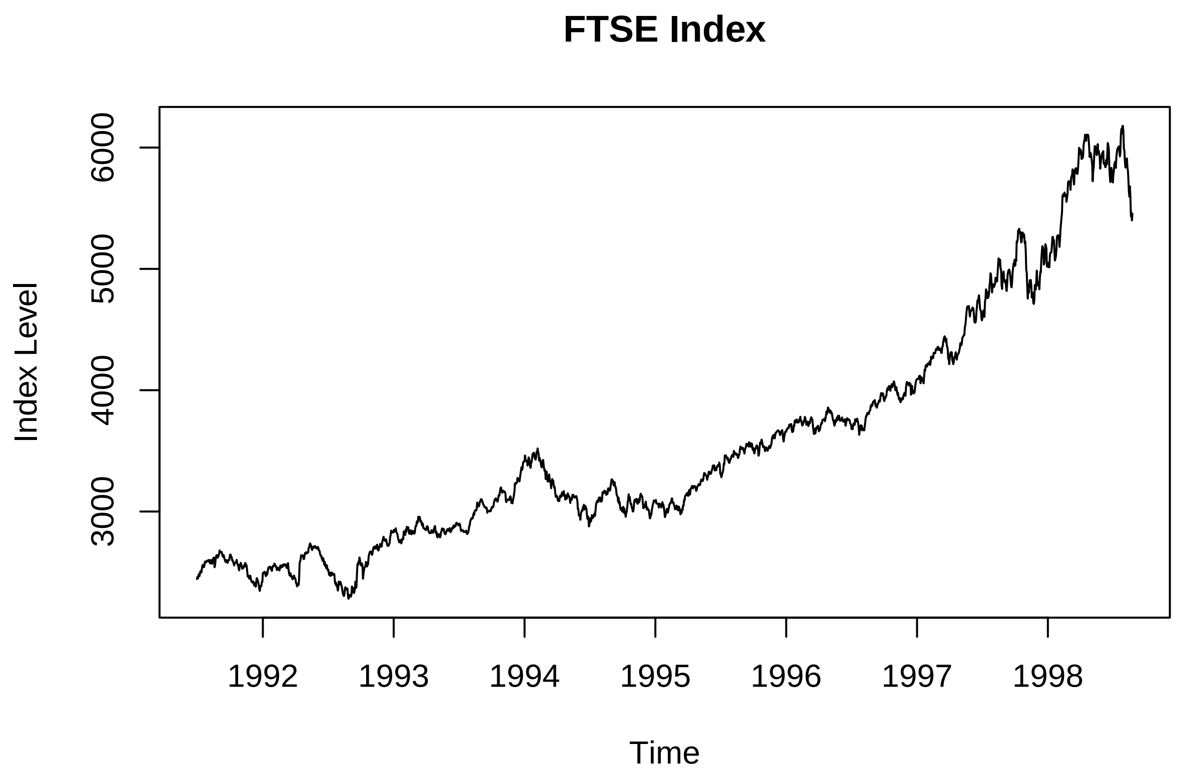




Regime-based modeling and long-run forecasting (FTSE)

We discretize FTSE log returns into 10 quantile bins, fit a 4-state HMM, and simulate the remaining path to estimate the final index value.

```r
ftse <- EuStockMarkets[, "FTSE"]
n <- length(ftse)
log_returns <- diff(log(ftse))
log_returns_q <- c(
  min(log_returns) * 1.01,
  quantile(log_returns, (1:9) / 10),
  max(log_returns) * 1.01
)
log_returns_cut <- cut(log_returns, log_returns_q)
levels(log_returns_cut) <- 1:10

ts.plot(ftse, main = "FTSE Index", ylab = "Index Level", xlab = "Time")
```



```
ftse[995:1005]
```

```
##  [1] 3199.9 3209.3 3214.9 3226.2 3217.6 3216.7 3220.4 3248.2 3262.6 3264.3
## [11] 3251.7
```

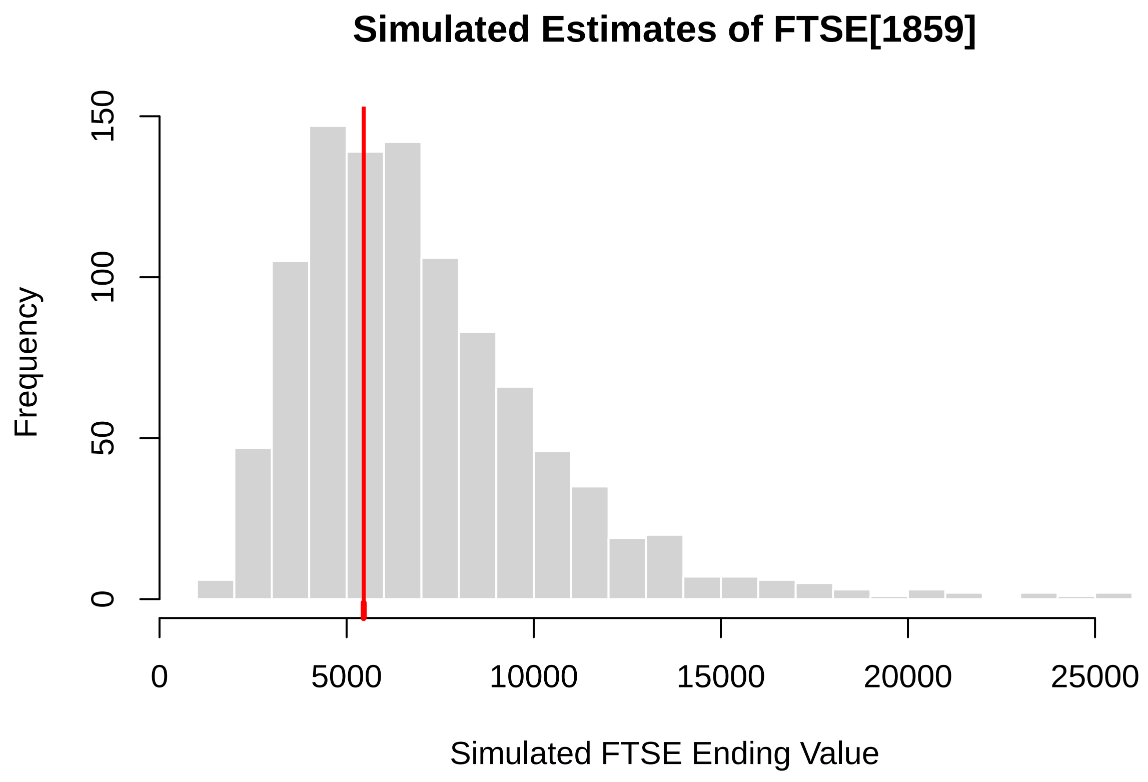
```
ftse[n]
```

```
## [1] 5455
```

```
P <- matrix(c(1:4, 4:1, 1, 4, 4, 1, 4, 1, 4, 1), nrow = 4, byrow = TRUE) / 10
PE <- matrix(
  c(1:10, 10:1, rep(5.5, 10), c(rep(1, 5), rep(10, 5))),
  nrow = 4,
  byrow = TRUE
) / 55

library(HMM)
states <- 1:4
symbols <- 1:10
hmm0 <- initHMM(
  States = states,
  Symbols = symbols,
  startProbs = rep(1 / 4, 4),
  transProbs = P,
  emissionProbs = PE
)

n_training <- 1000
```

```r
hmm_fit <- baumWelch(hmm0, observation = log_returns_cut[1:n_training])
```

```r
N <- 1000
n_testing <- n - n_training
index_859 <- numeric(N)

for (i in 1:N) {
  index <- as.numeric(simHMM(hmm_fit$hmm, n_testing)$observation)
  index_859[i] <- exp(
    cumsum(
      c(
        log(ftse[n_training]),
        runif(n_testing, min = log_returns_q[index], max = log_returns_q[index + 1])
      )
    )
  )[n_testing]
}

hist(
  index_859,
  breaks = 30,
  main = "Simulated Estimates of FTSE[1859]",
  xlab = "Simulated FTSE Ending Value",
  col = "lightgray",
  border = "white"
)
actual_val <- ftse[n]
rug(actual_val, col = "red", lwd = 3)
abline(v = actual_val, col = "red", lwd = 2)
```



```r
actual_val
```

```
## [1] 5455
```

The simulated values are realistic: the true final value lies within the dominant region of simulated outcomes, even with a heavy upper tail.

Option pricing (30-day horizon)

We price a European call option with strike 5000 when purchased 30 days before expiry.

```r
n_training <- 1829
n_testing <- 1859 - 1829
index_option <- numeric(N)

for (i in 1:N) {
  index <- as.numeric(simHMM(hmm_fit$hmm, n_testing)$observation)
  sim_log_ret <- runif(
    n_testing,
    min = log_returns_q[index],
    max = log_returns_q[index + 1]
  )
  sim_path <- exp(cumsum(c(log(ftse[n_training]), sim_log_ret)))
  index_option[i] <- sim_path[n_testing + 1]
}

K <- 5000
option_price <- mean((index_option - K) * (index_option > K))
```

```
option_price
```

```
## [1] 1142.766
```

With only 30 days to maturity, the expected payoff is much lower than longer-horizon pricing because there is less time for large upward moves.