Volatility forecasting: pick your own stock

Print-friendly PDF version

Choose a stock (or an index, a commodity or a cryptocurrency) that you find interesting, download its prices, and play with the volatility models from the lecture: look for the stylized facts, fit ARCH and GARCH models, and find out how well they forecast the volatility of your asset. The best results come when you compare notes with your neighbour, who has chosen another asset.

Pick an asset and download the data

The prices come from Yahoo Finance. Search for your asset on finance.yahoo.com and copy its *ticker*. Some examples:

Type	Tickers
US stocks	`AAPL`, `NVDA`, `TSLA`, `KO`
Oslo Børs (end with `.OL`)	`EQNR.OL`, `DNB.OL`, `NHY.OL`, `TEL.OL`, `YAR.OL`
Index	`^GSPC` (S&P 500)
Commodities	`CL=F` (oil), `GC=F` (gold)
Crypto	`BTC-USD`, `ETH-USD`

You need an asset with at least a few years of daily data.

```
install.packages(c("tidyquant", "rugarch", "xts"))
```

```
library(fpp3)
library(tidyquant)
library(xts)
library(rugarch)
```

Cheat sheet: `rugarch` syntax

Task	Syntax
Specify a GARCH(1,1) with Student's t errors	`ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)), mean.model = list(armaOrder = c(0, 0), include.mean = TRUE), distribution.model = "std")`
ARCH(r) and GARCH(r,s)	`garchOrder = c(r, 0)` and `garchOrder = c(r, s)`
Other variance models (optional part)	`model = "gjrGARCH"`, `model = "eGARCH"`
Other distributions	`distribution.model = "norm"`, `"std"` (Student's t), `"sstd"` (skewed t)

Task	Syntax
Fit	`ugarchfit(spec, data = x_train)` where `x_train` is an `xts` object or a numeric vector
Parameters, volatility and residuals	`coef(fit)`, `sigma(fit)`, `residuals(fit, standardize = TRUE)`
Information criteria, convergence	`infocriteria(fit)`, `convergence(fit)` (`0` means converged)
Diagnostic plots	`plot(fit)` (choose a plot with `which =`; 12 is the news impact curve)
Forecast	`ugarchforecast(fit, n.ahead = 10)`, plot with `plot(fc, which = 3)` (the volatility forecast); rolling: see the lecture notes (`n.roll`, `out.sample`), plot with `which = 4`
Quantile of a distribution	`qdist("std", p = 0.95, shape = coef(fit)["shape"])`

Step 0: Daily returns, and a test set

Download the daily prices from 2010, and calculate the daily log returns in percent from the *adjusted* price (which accounts for dividends and splits). Then split the returns into a **training set** and a **test set**: keep the last 250 trading days (about a year) as the test set. Use only the training data in Parts 1–3. The test set is for Part 4, where we check how well the models forecast data they have not seen.

Code hint

```
ticker <- "EQNR.OL"      # change this to your asset
prices <- tq_get(ticker, get = "stock.prices", from = as.Date("2010-01-01"))
ret <- prices %>%
  mutate(ret = 100 * (log(adjusted) - log(lag(adjusted)))) %>%
  filter(!is.na(ret)) %>%
  select(date, ret)
n_test <- 250                        # the last 250 trading days are the test set
ret_train <- ret %>% slice(1:(n() - n_test))
ret_test  <- ret %>% slice_tail(n = n_test)
x_train <- xts(ret_train$ret, order.by = ret_train$date)   # the format rugarch likes
x_test  <- xts(ret_test$ret,  order.by = ret_test$date)
```

Part 1: The stylized facts

Make a time plot of the (training) returns, the ACF of the returns and of the squared returns, and a histogram and a QQ-plot of the returns.

Code hint

```r
ret_train %>% ggplot(aes(x = date, y = ret)) + geom_line()
ret_ts <- ret_train %>% mutate(day = row_number()) %>% as_tsibble(index = day)
ret_ts %>% ACF(ret) %>% autoplot()
ret_ts %>% ACF(ret^2) %>% autoplot()
ggplot(ret_train, aes(x = ret)) +
  geom_histogram(aes(y = after_stat(density)), bins = 60) +
  stat_function(fun = dnorm, colour = "red",
                args = list(mean = mean(ret_train$ret), sd = sd(ret_train$ret)))
ggplot(ret_train, aes(sample = ret)) + geom_qq() + geom_qq_line()
ret_train %>% slice_max(abs(ret), n = 5)     # the five largest moves
```

Q: Which of the stylized facts (SF1–SF6 in the lecture) do you see for your asset? Find the dates of the largest moves: what happened? Compare with your neighbour: which of you has the more extreme tails?

Part 2: Fit a GARCH(1,1)

Fit a GARCH(1,1) with a constant mean to the training data, first with normal and then with Student's t innovations. Compare the two by their AIC and BIC. For the better model, calculate

- the persistence $\alpha_1 + \beta_1$,
- the unconditional standard deviation $\sqrt{\omega/(1-\alpha_1-\beta_1)}$, and the annualized version ($\times\sqrt{252}$; use 365 days for crypto).

Code hint

```r
spec_n <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "norm")
spec_t <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "std")
fit_n <- ugarchfit(spec_n, data = x_train)
fit_t <- ugarchfit(spec_t, data = x_train)
rbind(normal = infocriteria(fit_n)[1:2, 1], t = infocriteria(fit_t)[1:2, 1])
coef(fit_t)
pers <- unname(coef(fit_t)["alpha1"] + coef(fit_t)["beta1"])
c(persistence = pers,
  uncond_sd = sqrt(unname(coef(fit_t)["omega"]) / (1 - pers)))
```

Check the model: the standardized residuals $\hat{\varepsilon}_t = y_t/\hat{\sigma}_t$ should look like iid noise. Plot the ACF of the standardized residuals and of their squares. Then plot the estimated volatility $\hat{\sigma}_t$ together with the absolute returns.

Code hint

```
z <- as.numeric(residuals(fit_t, standardize = TRUE))
z_ts <- tibble(day = seq_along(z), z = z) %>% as_tsibble(index = day)
z_ts %>% ACF(z) %>% autoplot()
z_ts %>% ACF(z^2) %>% autoplot()
tibble(date = ret_train$date, ret = ret_train$ret, sigma = as.numeric(sigma(fit_t))) %>%
  ggplot(aes(x = date)) +
  geom_point(aes(y = abs(ret)), size = 0.3, colour = "grey60") +
  geom_line(aes(y = sigma), colour = "steelblue")
```

Q: How persistent is the volatility of your asset (is $\alpha_1 + \beta_1$ close to 1)? How many degrees of freedom does the t-distribution get, and what does that say about the tails? Did the GARCH remove the autocorrelation in the squared residuals?

Part 3: Play with the model

How many lags do you need? Fit an ARCH(1), an ARCH(5) and a GARCH(2,1) to your asset, all with Student's t innovations, and compare their AIC and BIC with the GARCH(1,1) from Part 2. Count the number of parameters of each model with `length(coef(fit))`.

Code hint

```
spec_arch1 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 0)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "std")
spec_arch5 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(5, 0)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "std")
spec_g21 <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(2, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "std")

fit_arch1 <- ugarchfit(spec_arch1, data = x_train)
fit_arch5 <- ugarchfit(spec_arch5, data = x_train)
fit_g21 <- ugarchfit(spec_g21, data = x_train)

rbind("ARCH(1)" = infocriteria(fit_arch1)[1:2, 1],
      "ARCH(5)" = infocriteria(fit_arch5)[1:2, 1],
      "GARCH(1,1)" = infocriteria(fit_t)[1:2, 1],
      "GARCH(2,1)" = infocriteria(fit_g21)[1:2, 1])
```

Q: Which model is best for your asset by AIC and by BIC? How many ARCH lags do you need to come close to the GARCH(1,1), and how many parameters does that cost? What does that tell you about why the GARCH model was introduced? Compare with your neighbour: do all assets behave the same way?

Part 4: Forecasting the volatility

Several days ahead. Forecast the volatility 20 days ahead from the end of your sample with the t-GARCH(1,1), and plot it. Compare the forecast for day 20 with the unconditional standard deviation from Part 2.

Code hint

```
fc20 <- ugarchforecast(fit_t, n.ahead = 20)
plot(fc20, which = 3)                                  # the forecast of sigma
sigma(fc20)[20]                                        # the forecast for day 20
sqrt(unname(coef(fit_t)["omega"]) / (1 - pers))        # the unconditional sd
```

Q: Is the volatility of your asset today above or below its long-run level? What happens to the forecast as the horizon grows, and how fast?

Out of sample. Now the test set comes in. Fix the parameters of the t-GARCH(1,1) that you estimated on the training data, and make one-step-ahead forecasts of the volatility for each of the 250 test days (this is what the `n.roll` and `out.sample` arguments of `ugarchforecast` are for, see the in-class notes). The forecasts need the observed test returns to update the volatility, so give `ugarchforecast` the training and the test data together. From the forecasts, make 90% prediction intervals for the returns, and check how many of the test returns that fall inside.

Code hint

```
spec_fixed <- getspec(fit_t); setfixed(spec_fixed) <- as.list(coef(fit_t))
x_all <- rbind(x_train, x_test)     # the forecasts need the observed test returns
fc <- ugarchforecast(spec_fixed, data = x_all, n.ahead = 1,
                     n.roll = n_test - 1, out.sample = n_test)
test <- ret_test %>%
  mutate(sigma = as.numeric(sigma(fc)),
         mu = as.numeric(fitted(fc)))
q <- qdist("std", 0.95, shape = coef(fit_t)["shape"])
test <- test %>% mutate(lower = mu - q * sigma, upper = mu + q * sigma)
mean(test$ret >= test$lower & test$ret <= test$upper)       # coverage
ggplot(test, aes(x = date)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "steelblue", alpha = 0.25) +
  geom_point(aes(y = ret), size = 0.7)
```

Does GARCH beat the simple alternatives? Compare your GARCH forecasts with two benchmarks that do not use GARCH: the *constant* standard deviation from the training

data, and the standard deviation of the *last 20 returns*. For each method, make 90% prediction intervals for the returns, and calculate

- the **coverage** of the intervals: the share of the test returns inside them (should be about 90%),
- the **Winkler score** of the intervals (from the lecture on forecast accuracy): the width of the interval, plus a penalty of 20 times the distance to the interval for returns outside it. Smaller is better,
- the mean squared error of the variance forecast, compared with the squared return (which is a noisy measure of the variance). Smaller is better.

Code hint

```
ret_all <- bind_rows(ret_train, ret_test)
sd20 <- dplyr::lag(slider::slide_dbl(ret_all$ret, sd, .before = 19, .complete = TRUE))
test <- test %>%
  mutate(sig_const = sd(ret_train$ret),      # constant sd (training data)
         sig_roll = tail(sd20, n_test))      # sd of the previous 20 returns
e <- test$ret - test$mu
evaluate <- function(sigma, q) {
  lower <- test$mu - q * sigma
  upper <- test$mu + q * sigma
  penalty <- 20 * pmax(lower - test$ret, 0) + 20 * pmax(test$ret - upper, 0)
  c(MSE = mean((e^2 - sigma^2)^2),
    coverage_90 = mean(test$ret >= lower & test$ret <= upper),
    Winkler = mean((upper - lower) + penalty))
}
rbind("GARCH" = evaluate(test$sigma, q),
      "Constant sd" = evaluate(test$sig_const, qnorm(0.95)),
      "Rolling 20-day sd" = evaluate(test$sig_roll, qnorm(0.95)))
```

Q: Which method gives the best forecasts of the volatility? Is the coverage of the GARCH intervals close to 90%? What happens to the constant-sd intervals if the test period is calmer or more turbulent than the training period?

> **i** Discuss with the person next to you
>
> **What did your neighbour get for a different stock?** Compare your results from Parts 2–4:
>
> - the persistence $\alpha_1 + \beta_1$ and the degrees of freedom of the t-distribution: whose volatility has the longest memory and the heaviest tails?
> - the best model in Part 3: was it the same for both of you, and how close was the race?
> - Part 4: did GARCH beat the constant and rolling standard deviations for both of you? Whose constant-sd intervals were too narrow, and whose were too wide?
>
> Why do the results differ between your assets? Think about what happened to each of them in the training period and in the test year.

Part 5: Go further (optional)

More advanced models. There are many extensions of the GARCH model. Try these three on your asset:

- the **GJR-GARCH**, which lets *negative* returns increase the volatility more than positive ones (the *leverage effect*), with parameter `gamma1`,
- the **EGARCH**, which models the logarithm of the variance,
- the GARCH(1,1) with a **skewed** t distribution.

Compare their AIC and BIC with the GARCH(1,1). For the GJR-GARCH, look at `gamma1` and at the news impact curve, which shows the volatility tomorrow as a function of the return today. Some of these models can be hard to estimate: check that `convergence(fit)` is 0 before you trust a model.

Code hint

```
spec_gjr <- ugarchspec(
  variance.model = list(model = "gjrGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "std")
spec_egarch <- ugarchspec(
  variance.model = list(model = "eGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "std")
spec_sstd <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "sstd")

fit_gjr <- ugarchfit(spec_gjr, data = x_train)
fit_egarch <- ugarchfit(spec_egarch, data = x_train)
fit_sstd <- ugarchfit(spec_sstd, data = x_train)

convergence(fit_egarch)      # 0 means that the optimizer converged

rbind("GARCH(1,1)" = infocriteria(fit_t)[1:2, 1],
      "GJR-GARCH(1,1)" = infocriteria(fit_gjr)[1:2, 1],
      "EGARCH(1,1)" = infocriteria(fit_egarch)[1:2, 1],
      "GARCH(1,1), skewed t" = infocriteria(fit_sstd)[1:2, 1])

coef(fit_gjr)                # gamma1: the leverage effect
plot(fit_gjr, which = 12)    # the news impact curve
```

Q: Is there a leverage effect for your asset (a positive `gamma1`, and a news impact curve that is steeper on the left)? Does any of the advanced models beat the GARCH(1,1) clearly?

Other ideas. Pick one or more of these, and report what you find to the class:

1. **Value at risk.** The 1% value at risk is the return we expect to fall below on 1% of the days: $\hat{\mu}_t + q_{0.01}\,\hat{\sigma}_t$, with `q <- qdist("std", p = 0.01, shape = ...)`. Count the exceedances in the test set: with 250 days we expect about 2.5. Do the same for the 5% level.
2. **A different period.** Estimate the model on 2010–2019 only, and look at what the model predicted for the volatility in March 2020. How quickly did the forecast catch up?
3. **Re-estimation.** Re-estimate the model every month instead of keeping the parameters fixed (`ugarchroll` does this). Does it help?
4. **A different asset.** Compare the persistence for a stock, an index and a cryptocurrency. Which has the longest memory?
5. **A mean model.** Add ARMA terms to the mean (`armaOrder = c(1, 1)`). Does it improve the fit or the forecasts of the returns themselves?
6. **Estimation window.** Estimate on the last two, five and ten years only. Do the parameters change?