The University of Queensland Australia | CREATE CHANGE

ECON3350 - Applied Econometrics for Macroeconomics and Finance

Tutorial 6: Modelling Volatility – II

Tutor: Francisco Tavares Garcia

Report 1 is due today at 1 pm!

Instructions

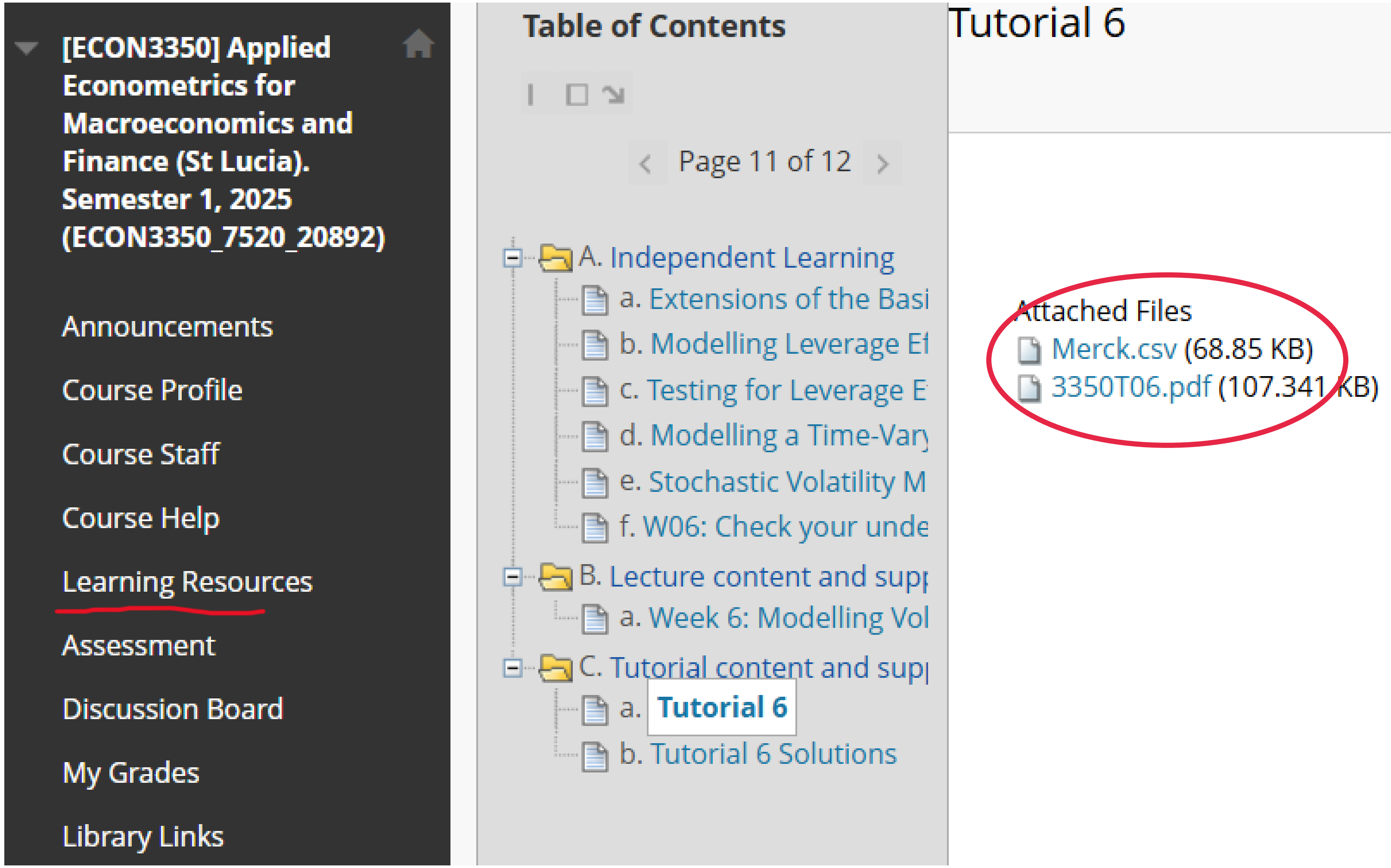
The project consists of three research questions. Please answer all questions as clearly, concisely and completely as possible. Each question is worth 50 marks, for a total of 150 marks. This report will constitute 20% of your overall grade in this course.

We suggest that you use R for all empirical work involved. However, you should be able to use another statistical software (e.g. Eviews, Stata, Python, etc.) without a problem. If you do choose to work with an alternative software, please note that support for software-specific issues from the course coordinator and tutors may be very limited.

Please upload your report via the "Turnitin" submission link (in the "Assessment / Research Report" folder). Please note that hard copies *will not* be accepted. At the moment, the due date is 1:00 PM on 28 March 2025, but please check BlackBoard regularly for announcements regarding any changes to this. Your report should be a write-up of your answers (in PDF format, single-spaced, and in 12 font size).[1]

You are allowed to work on this assignment with others, i.e., you can discuss how to answer the questions with your classmate(s) and even use AI as a research tool. However, this is not a group assignment, which means that the report must be written individually and by you: you must answer all the questions in your own words and submit your report separately. The marking system will check for similarities and AI content, and UQ's student integrity and misconduct policies on plagiarism *strictly apply*.

Let's download the tutorial and the dataset.



Now, let's download the script for the tutorial.

- Copy the code from Github,
  - https://github.com/tavaresgarcia/teaching
- Save the scripts in the same folder as the data.

Tutorial 6: Modelling Volatility - II

At the end of this tutorial you should be able to:

- construct an adequate set of models with possible TGARCH errors;
- construct an adequate set of models with possible GARCH-in-mean components;
- use R to draw inference on the presence of leverage effects from a class of TGARCH models;
- use R to draw inference on the presence of time-varying risk premia from a class of GARCH-in-mean models;
- use R to estimate and forecast volatilities based on models with TGARCH errors or GARCH-in-mean components.

Recap – last tutorial

In the (invertible) ARMA model:

$$y_t = a_0 + \sum_{j=1}^{p} a_1 y_{t-j} + \sum_{j=1}^{q} b_q \varepsilon_{t-j} + \varepsilon_t, \qquad \varepsilon_t \equiv \nu_t h_t^{\frac{1}{2}}$$

where $\nu_t \sim \mathcal{N}(0,1)$ and
$\mathsf{E}(\varepsilon_t \nu_{t-s}) = 0$ for all t, s.

we assumed $\mathsf{Var}(\varepsilon_t \,|\, y_{t-1}, y_{t-2}, \dots) = \sigma^2$: this is a linear model.

ARCH(q_h)

Looks like an MA(1)

$$\varepsilon_t^2 = \nu_t^2 h_t, \quad h_t \equiv \mathsf{E}(\varepsilon_t^2 \,|\, \varepsilon_{t-1}, \varepsilon_{t-2}, \dots) = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2.$$

H(q_h)

GARCH(p_h, q_h)

$$\varepsilon_t^2 = \nu_t^2 h_t,$$

$$h_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j \varepsilon_{t-j}^2 + \sum_{j=1}^{p} \beta_j h_{t-j},$$

Autoregressive lags of h_t.

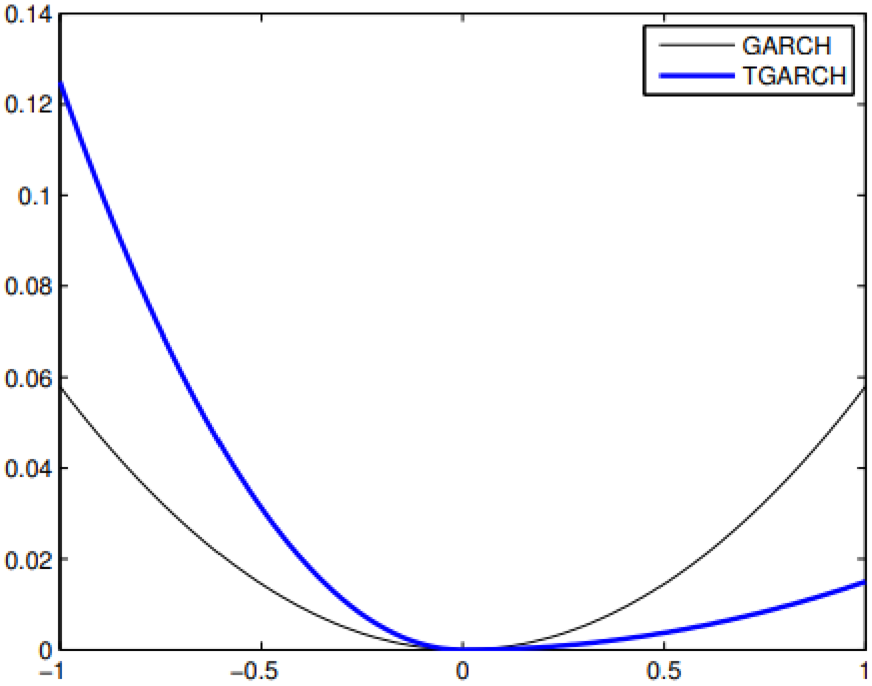
Looks like an ARMA(p,q)

$$= \alpha_0 + \alpha(L)\varepsilon_{t-1}^2 + \beta(L) h_{t-1},$$

In practice, the most commonly used is the GARCH$(1,1)$:

$$h_t = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 h_{t-1}.$$

This tutorial Q1 – The threshold GARCH (TGARCH)



The TGARCH or GJR Model

The threshold GARCH (TGARCH) model (developed by Glosten, *et al.*, 1994) is

$$h_t = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \lambda d_{t-1} \varepsilon_{t-1}^2 + \beta_1 h_{t-1},$$

$$d_{t-1} = \begin{cases} 1 & \text{if } \varepsilon_{t-1} < 0, \\ 0 & \text{otherwise.} \end{cases}$$

We require $\alpha_0 \geq 0$ and $\alpha_1 \geq 0$ for non-negativity.

The effect of ε_{t-1} on h_t:

- if $\varepsilon_{t-1} \geq 0$, $d_{t-1} = 0$ and the effect is $\alpha_1 \varepsilon_{t-1}^2$;
- if $\varepsilon_{t-1} < 0$, $d_{t-1} = 1$ and the effect is $(\alpha_1 + \lambda)\varepsilon_{t-1}^2$;
- interpreted as leverage effect when $\lambda > 0$.

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

Problems

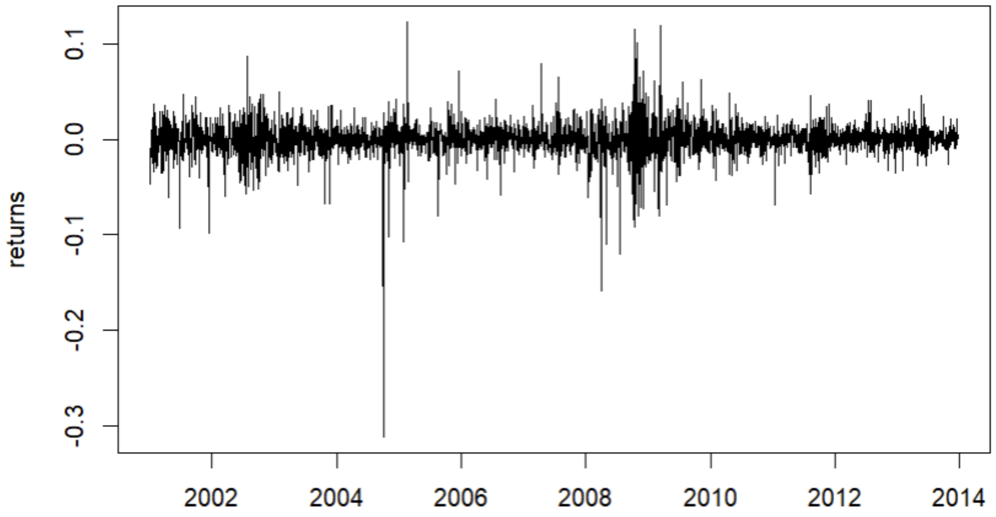
Consider the daily share prices of Merck & Co., Inc. (MRK) for the period 2 January 2001 to 23 December 2013 in the data file `Merck.csv`. Let $\{y_t\}$ denote the process of share prices. Recall that we learned how to fit ARMA-ARCH/GARCH models to data last week. We now consider extensions of these models to capture possible leverage effects and time-varying risk premia.

Solution For this tutorial, we load the following useful packages.

```
library(forecast)
library(dplyr)
library(rugarch)
```

Next, load the data, extract the variables and plot them to get an idea on what we are working with.

```
mydata <- read.delim("Merck.csv", header = TRUE,  sep = ",")

date <- as.Date(mydata$Date, format = "%d/%m/%Y")
y <- mydata$y
r <- diff(log(y))

plot(date[-1], r, type = "l", xlab = "", ylab = "returns")
```



1. Consider the class of ARMA(p_m, q_m)-TGARCH(p_h, q_h) models for the returns $r_t = \ln y_t - \ln y_{t-1}$.

(a) Construct an adequate set of models for estimating and forecasting volatilities.

To construct and adquate set of models, we use a similar approach to the one developed in Week 8 for the class of basic GARCH models. The additional dimension considered here is whether the model includes the "threshold term''.

Note that the `rugarch` package has a different naming convention for threshold GARCH models. Specifically, what we refer to as TGARCH, is called the gjr-GARCH in the package, whereas TGARCH denotes a slightly different model. This is not uncommon, so it is always important to check the documentation of any particular software / package carefully!

```
submods <- c("sGARCH", "gjrGARCH")
ARMA_TGARCH_est <- list()
ic_arma_tgarch <- matrix( nrow = 4 ^ 2 * 2 ^ 3, ncol = 7 )
colnames(ic_arma_tgarch) <- c("pm", "qm", "ph", "qh",
                              "thresh", "aic", "bic")
i <- 0; t0 <- proc.time()
for (pm in 0:3)
{
  for (qm in 0:3)
  {
    for (ph in 0:1)
    {
      for (qh in 0:1)
      {
        for (thresh in 0:1)
```

```
        {
          i <- i + 1
          ic_arma_tgarch[i, 1:5] <- c(pm, qm, ph, qh, thresh)

          if (ph == 0 && qh == 0)
          {
            # no such thing as a homoscedastic threshold ARMA!
            if (!thresh)
            {
              # for models with constant variance, the ugarchspec
              # and ugarchfit functions do not work well;
              # instead, the documentation advises to use
              # arfimaspec and arfimafit
              try(silent = T, expr =
              {
                ARMA_TGARCH_mod <- arfimaspec(
                  mean.model = list(armaOrder = c(pm, qm)))

                ARMA_TGARCH_est[[i]] <- arfimafit(
                                              ARMA_TGARCH_mod, r)

                ic_arma_tgarch[i,6:7] <- infocriteria(
                                         ARMA_TGARCH_est[[i]])[1:2]
```

```
              })
            }
          }
          else
          {
            try(silent = T, expr =
            {
              ARMA_TGARCH_mod <- ugarchspec(
                mean.model = list(armaOrder = c(pm, qm)),
                variance.model = list(model = submods[1 + thresh],
                                      garchOrder = c(ph, qh)))

              ARMA_TGARCH_est[[i]] <- ugarchfit(ARMA_TGARCH_mod, r,
                                                 solver = 'hybrid')

              ic_arma_tgarch[i,6:7] <- infocriteria(
                                      ARMA_TGARCH_est[[i]])[1:2]
            })
          }
        }
      }
    }
  }
}
cat("\n", proc.time() - t0, "\n")
```

```
##
##  142.42 0.18 144.18 NA NA
```

```
ic_aic_arma_tgarch <- ic_arma_tgarch[
                              order(ic_arma_tgarch[,6]),][1:20,]
ic_bic_arma_tgarch <- ic_arma_tgarch[
                              order(ic_arma_tgarch[,7]),][1:20,]

ic_int_arma_tgarch <- intersect(as.data.frame(ic_aic_arma_tgarch),
                                as.data.frame(ic_bic_arma_tgarch))

adq_set_arma_tgarch <- as.matrix(arrange(as.data.frame(
                    ic_int_arma_tgarch), pm, qm, ph, qh, thresh))
adq_idx_arma_tgarch <- match(data.frame(
                          t(adq_set_arma_tgarch[, 1:5])),
                          data.frame(t(ic_arma_tgarch[, 1:5])))
```

Note that we have selected the entire *intersecting* set of models. The next step is to have a look at residual autocorrelation. As with the basic GARCH, one must be careful in interpreting the Ljung-Box test with conditional heteroscedasticity. Indeed, we will avoid it altogether and instead just examine the SACF.

```
nmods <- length(adq_idx_arma_tgarch)
sacf_tgarch <- matrix(nrow = nmods, ncol = 15)
colnames(sacf_tgarch) <- c("pm", "qm", "ph", "qh", "thresh", 1:10)
for (i in 1:nmods)
{
  sacf_tgarch[i,1:5] <- adq_set_arma_tgarch[i,1:5]
  sacf_tgarch[i,6:15] <-
              acf(ARMA_TGARCH_est[[adq_idx_arma_tgarch[i]]]@fit$z,
                                          lag = 10, plot = F)$acf[2:11]
}
```

All the residuals look relatively small, so we do not eliminate any specifications from the existing set of models. Hence, this is our adequate set.

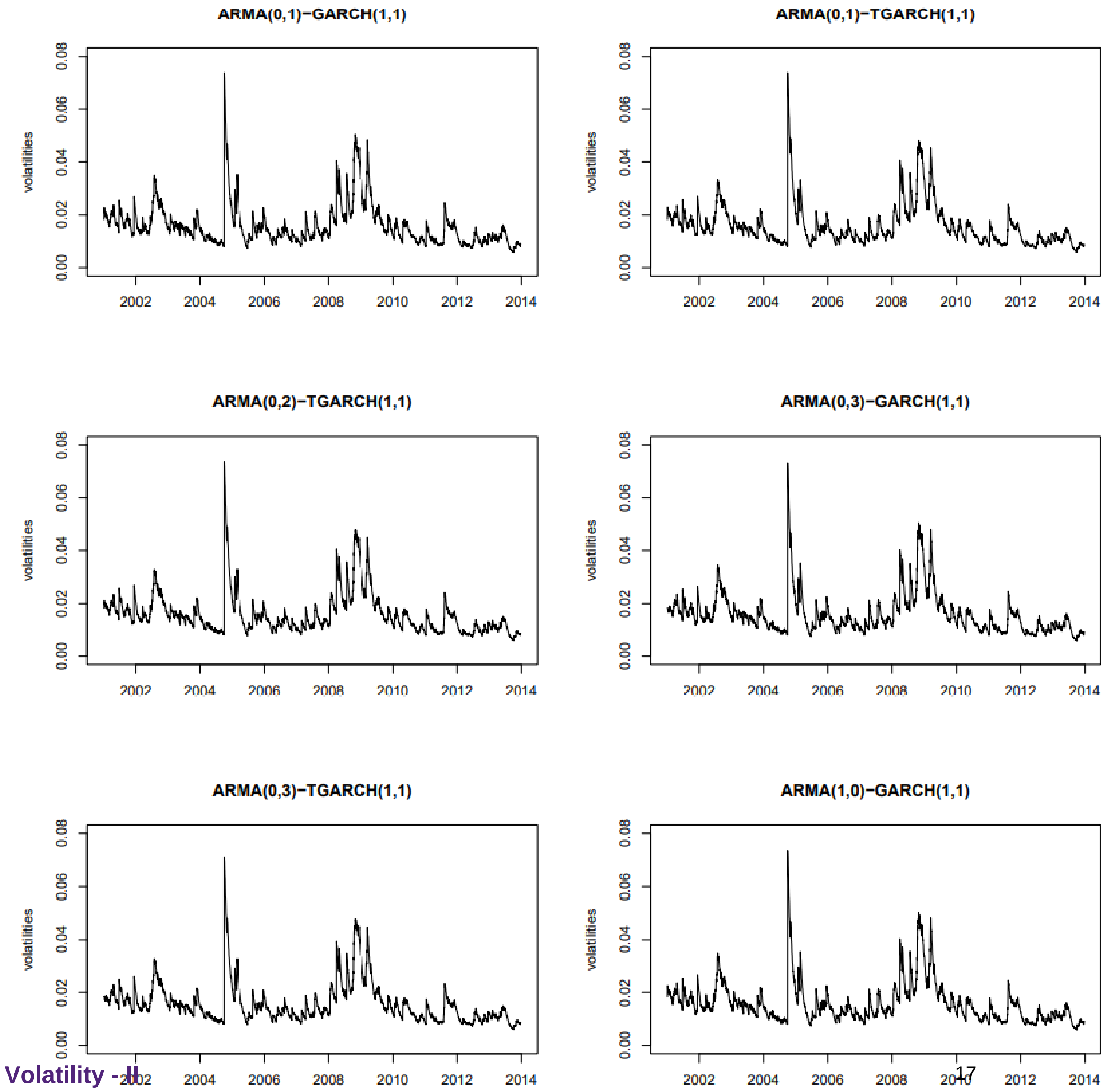
	pm	qm	ph	qh	thresh	1	2	3	4	5	6	7	8	9	10
1	0	0	1	1	0	0.0081936479	-0.008180346	-0.020876116	0.01901508	-0.008910533	-0.004774759	-0.012078954	0.01485534	0.01478805	0.02010951
2	0	0	1	1	1	0.0083977072	-0.009818170	-0.020576030	0.01727924	-0.009946407	-0.004805960	-0.010267812	0.01335302	0.01422120	0.02119350
3	0	1	1	1	0	0.0158449116	-0.009494245	-0.020898314	0.01879817	-0.008575444	-0.004841945	-0.010958642	0.01371352	0.01469286	0.02034625
4	0	1	1	1	1	0.0144209032	-0.008725690	-0.020220719	0.01729106	-0.009781031	-0.004942526	-0.011168642	0.01466784	0.01519935	0.02115653
5	0	2	1	1	1	0.0062150667	-0.002712831	-0.020421168	0.01768976	-0.010018919	-0.004577504	-0.011302468	0.01487144	0.01457072	0.02122713
6	0	3	1	1	0	0.0016254622	-0.009127677	-0.008551879	0.01962796	-0.008208408	-0.004546545	-0.010288477	0.01323325	0.01355036	0.01988715
7	0	3	1	1	1	0.0119037768	-0.003162606	0.003090857	0.01838743	-0.008568439	-0.004151910	-0.010102439	0.01431051	0.01476632	0.02081990
8	1	0	1	1	0	0.0103544183	-0.009409442	-0.021051053	0.01903990	-0.008648679	-0.004712717	-0.011038853	0.01367091	0.01439632	0.02024350
9	1	0	1	1	1	0.0144050579	-0.008779628	-0.020227445	0.01730724	-0.009770969	-0.004947657	-0.011186022	0.01467302	0.01519917	0.02114058
10	1	1	1	1	0	0.0125788127	-0.006280339	-0.018763483	0.02037804	-0.007427800	-0.003929927	-0.010547775	0.01435935	0.01481264	0.02043372
11	1	2	1	1	0	0.0006729399	0.004996916	-0.012690374	0.02543645	-0.004623293	-0.001337389	-0.008813224	0.01430292	0.01389760	0.02057476
12	1	3	1	1	1	0.0198209579	0.002596766	0.005681992	0.03528817	0.004296872	0.005250292	-0.001876993	0.01870269	0.01793991	0.02368950
13	3	0	1	1	1	0.0019700921	-0.008076868	0.009128618	0.01857870	-0.008822545	-0.004987557	-0.009697494	0.01368863	0.01420221	0.02053189
14	3	1	1	1	1	0.0112024444	-0.007023111	-0.006910362	0.01807763	-0.009192444	-0.005078630	-0.010460940	0.01436635	0.01466940	0.02084163

(b) Draw inference on historic volatilities.

Solution Plotting estimated volatilities is the same as for the basic GARCH (note that we still do not get confidence intervals with the `rugarch` package).

```
title_tgarch <- rep(NA, times = nmods)
for (i in 1:nmods)
{
  title_tgarch[i] <- paste("ARMA(",
                           as.character(adq_set_arma_tgarch[i, 1]), ",",
                           as.character(adq_set_arma_tgarch[i, 2]),
                           ")-",
                           c("", "T")[1 + adq_set_arma_tgarch[i,5]],
                           "GARCH(",
                           as.character(adq_set_arma_tgarch[i, 3]), ",",
                           as.character(adq_set_arma_tgarch[i, 4]), ")",
                           sep = "")
  plot(date[-1], sqrt(
        ARMA_TGARCH_est[[adq_idx_arma_tgarch[i]]]@fit$var),
        type = "l", xlab = "", ylab = "volatilities",
        ylim = c(0, 0.08), main = title_tgarch[i])
}
```

All volatility estimates generally look very similar across the specifications in the adequate set. TGARCH vs GARCH specifications generally do not produce noticeable differences for fixed p_m, q_m, p_h and q_h. However, larger ARMA lags (p_m and q_m) appear to be associated with larger “spikes’’ in volatilities when they do occur.



(c) Draw inference on the existence of leverage effects.

Testing for Leverage Effects with EGARCH or TGARCH

Recall that in both the EGARCH and TGARCH, leverage effects are controlled by a single parameter λ.

If either model is estimated, only a simple one-tailed t-test on λ is needed to test for leverage effects:

$H_0 : \lambda = 0,$

$H_1 : \lambda < 0$ for the EGARCH or $\lambda > 0$ for the TGARCH.

(c) Draw inference on the existence of leverage effects.

Solution To test for leverage effects, we focus on on the threshold parameter in models where the threshold is actually included. Note that the `rugarch` labels the threshold parameter γ, whereas we used the notation λ in the videos.

```
for (i in 1:nmods)
{
  if (adq_set_arma_tgarch[i, 5] == 1)
  {
    lambda_est <- ARMA_TGARCH_est[[
              adq_idx_arma_tgarch[i]]]@fit$coef["gamma1"]
    lambda_tvl <- ARMA_TGARCH_est[[
              adq_idx_arma_tgarch[i]]]@fit$tval["gamma1"]
    cat(paste0(title_tgarch[i], ": lambda_hat = ",
                          round(lambda_est, 2),
                          ", t-value = ",
                          round(lambda_tvl, 2)),
                          "\n")
  }
}
```

```
## ARMA(0,0)-TGARCH(1,1): lambda_hat = 0.02, t-value = 4045.69
## ARMA(0,1)-TGARCH(1,1): lambda_hat = 0.01, t-value = 15801.59
## ARMA(0,2)-TGARCH(1,1): lambda_hat = 0.01, t-value = 23628.55
## ARMA(0,3)-TGARCH(1,1): lambda_hat = 0.01, t-value = 50575.42
## ARMA(1,0)-TGARCH(1,1): lambda_hat = 0.01, t-value = 16991.77
## ARMA(1,2)-TGARCH(1,1): lambda_hat = 0.01, t-value = 3346.59
## ARMA(1,3)-TGARCH(1,1): lambda_hat = 0.03, t-value = 4202.3
## ARMA(3,0)-TGARCH(1,1): lambda_hat = 0.01, t-value = 4184.12
## ARMA(3,1)-TGARCH(1,1): lambda_hat = 0.01, t-value = 11655.48
```

The null $H_0 : \lambda = 0$ is easily rejected in favour of $H_1 : \lambda > 0$ for all specifications at a very low significance level. Hence, we may infer that there is substantial evidence of "leverage effects".

It is worth pointing out that we can justify this conclusion based on the hypothesis test in TGARCH specifications only. The fact that some GARCH specifications are also included in the adequate is *irrelevant* for this purpose. The latter simply means that we cannot eliminate basic GARCH models on the basis of *fit vs parsimony* trade-off along with risk of residual autocorrelation.

$$h_t = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \lambda d_{t-1} \varepsilon_{t-1}^2 + \beta_1 h_{t-1},$$

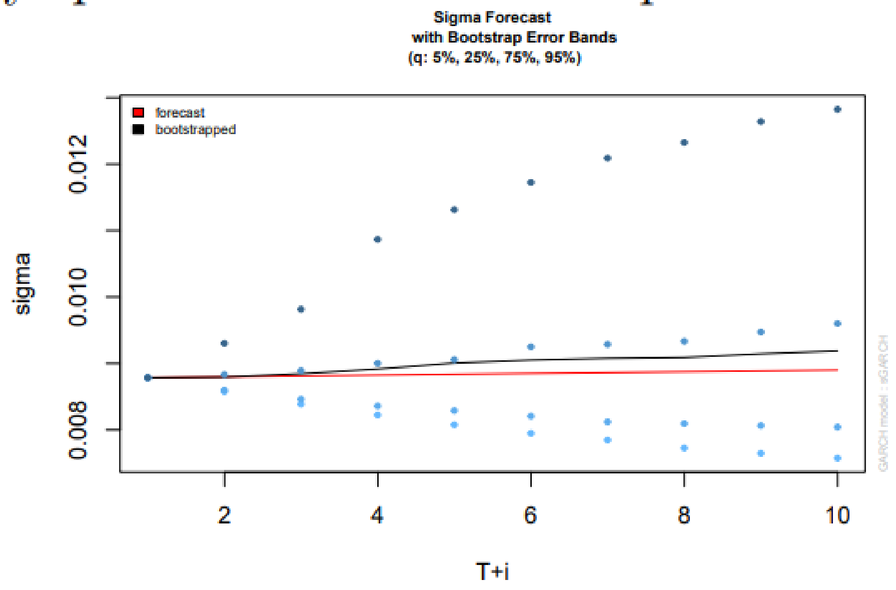
$$d_{t-1} = \begin{cases} 1 & \text{if } \varepsilon_{t-1} < 0, \\ 0 & \text{otherwise.} \end{cases}$$

(d) Forecast volatility for the four trading days past the end of the sample.

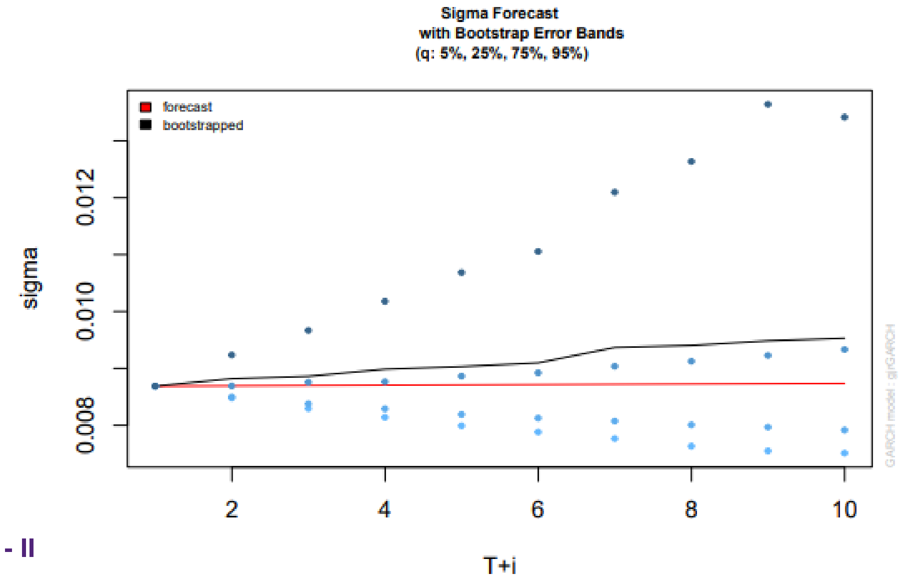
Solution Volatility forecasts are generated using the `ugarchboot` function, which also provides *partial* predictive intervals—i.e., that account for uncertainty related to unknown future shocks, but *not* estimation uncertainty. In fact, `ugarchboot` can also be instructed to compute predictive intervals that account for estimation uncertainty by specifying option `method = "Full"`, but this is computationally very time consuming, so we do not attempt it in this exercise.

Also, note that `rugarch` provides a special version of the `plot` function that will plot the volatility forecasts along with predictive intervals. Unfortunately, the functionality of this customised function is limited: it does not allow the user to modify the plot title, axis limits, etc.

```
for (i in 1:nmods)
{
  plot(ugarchboot(ARMA_TGARCH_est[[adq_idx_arma_tgarch[i]]],
                            method = "Partial"), which = 3)
}
```



There are subtle differences between the volatility forecasts generated by different GARCH and TGARCH specifications, but all agree that volatilities across the forecast horizon will remain well below the levels estimated around 2005 and then again in 2008/2009.



2. Consider a class of GARCH-M models for returns r_t, with one ARCH lag and one GARCH lag.

(G)ARCH-in-Mean

We expect risk to be compensated by a higher expected return; so why not let the expected return be partly determined by risk?

Engle, *et al.* (1987) suggested the ARCH-in-Mean, or ARCH-M, specification:

$$y_t = \mu_t + \varepsilon_t,$$

$$\mu_t = \beta + \delta\sqrt{h_t} = \beta + \delta\sqrt{\alpha_0 + \alpha_1 \sum_{j=1}^{q} w_j \varepsilon_{t-j}^2},$$

where α_0, α_1, β and $\delta > 0$ are constants; $w_1, \ldots, w_q$ are weights assigned to past q squared errors (e.g., set $w_j = (5-j)/10$ for $j = 1, \ldots, 4$).

2. Consider a class of GARCH-M models for returns r_t, with one ARCH lag and one GARCH lag.

(a) Construct an adequate set of models for estimating the risk-premium.

Solution We use the same approach as with GARCH/TGARCH models; here it is some what simpler because we fix all GARCH lags to be 1. So a specification consists of ARMA lags p, q and an indicator as whether or not the GARCH-in-mean term is included.

```
ARMA_GARCHM_est <- list()
ic_arma_garchm <- matrix( nrow = 4 ^ 2 * 2, ncol = 5 )
colnames(ic_arma_garchm) <- c("p", "q", "m", "aic", "bic")
i <- 0; t0 <- proc.time()
for (p in 0:3)
{
  for (q in 0:3)
  {
    for (m in 0:1)
    {
      i <- i + 1
      ic_arma_garchm[i, 1:3] <- c(p, q, m)
      try(silent = T, expr =
      {
        ARMA_GARCHM_mod <- ugarchspec(
                  mean.model = list(armaOrder = c(p, q),
                                    archm = m),
                  variance.model = list(garchOrder = c(1, 1)))

        ARMA_GARCHM_est[[i]] <- ugarchfit(ARMA_GARCHM_mod, r,
                                          solver = 'hybrid')

        ic_arma_garchm[i,4:5] <- infocriteria(
                                  ARMA_GARCHM_est[[i]])[1:2]
      })
    }
  }
}
cat("\n", proc.time() - t0, "\n")
```

```
##
##  36.33 0 36.42 NA NA
```

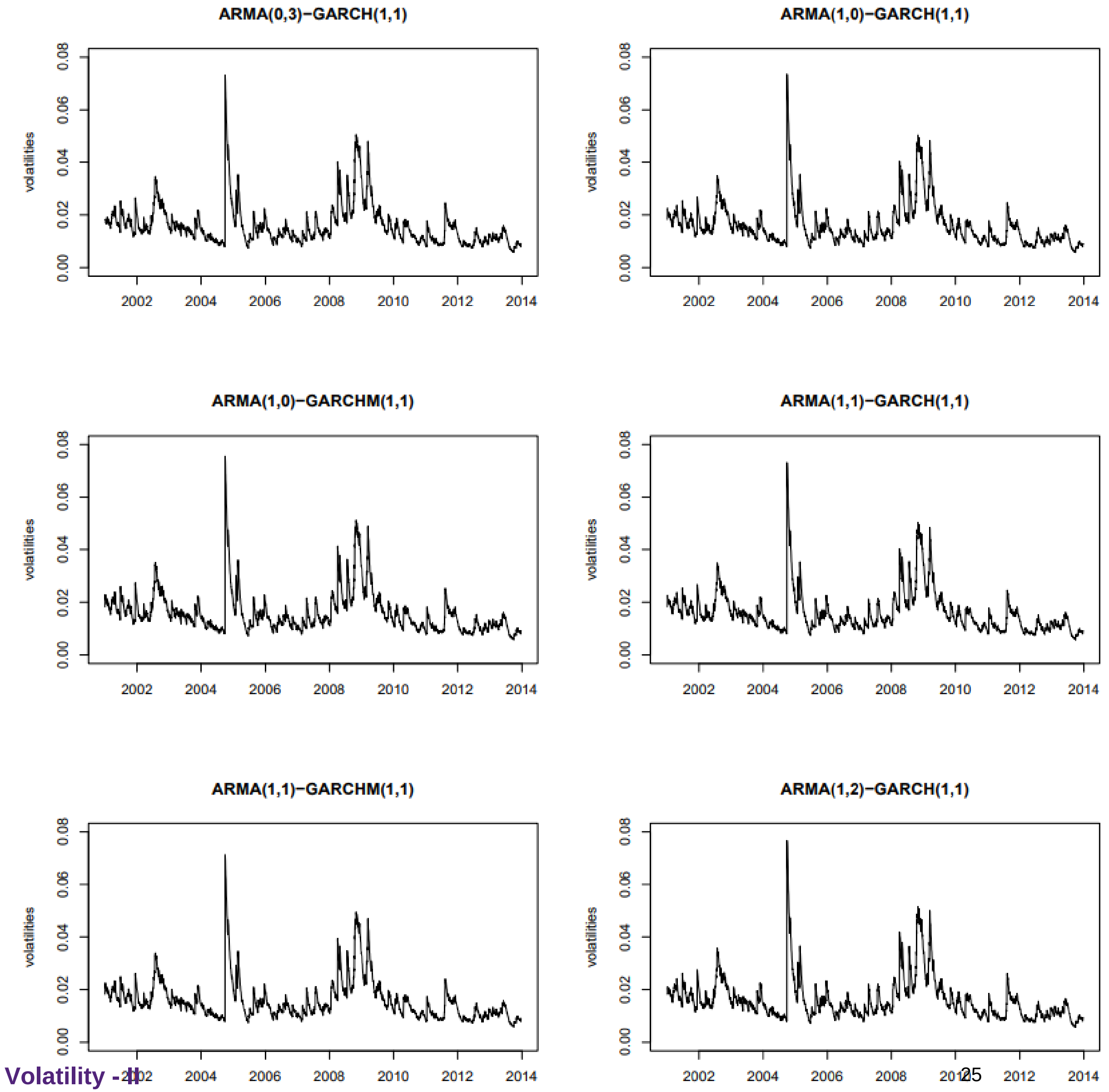
```r
ic_aic_arma_garchm <- ic_arma_garchm[
                        order(ic_arma_garchm[,4]),][1:20,]
ic_bic_arma_garchm <- ic_arma_garchm[
                        order(ic_arma_garchm[,5]),][1:20,]

ic_int_arma_garchm <- intersect(
                        as.data.frame(ic_aic_arma_garchm),
                        as.data.frame(ic_bic_arma_garchm))

adq_set_arma_garchm <- as.matrix(arrange(as.data.frame(
                            ic_int_arma_garchm), p, q, m))
adq_idx_arma_garchm <- match(
                     data.frame(t(adq_set_arma_garchm[, 1:3])),
                     data.frame(t(ic_arma_garchm[, 1:3])))

nmods <- length(adq_idx_arma_garchm)
sacf_garchm <- matrix(nrow = nmods, ncol = 13)
colnames(sacf_garchm) <- c("p", "q", "m", 1:10)
for (i in 1:nmods)
{
  sacf_garchm[i,1:3] <- adq_set_arma_garchm[i,1:3]
  sacf_garchm[i,4:13] <-
        acf(ARMA_GARCHM_est[[adq_idx_arma_garchm[i]]]@fit$z,
                                lag = 10, plot = F)$acf[2:11]
}
```

	p	q	m	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0.0081936479	-0.008180346	-0.020876116	0.01901508	-0.008910533	-0.004774759	-0.012078954	0.01485534	0.01478805	0.02010951
2	0	0	1	0.0087034897	-0.008587421	-0.020796462	0.01936015	-0.008721620	-0.004351718	-0.011162801	0.01436324	0.01480487	0.02046914
3	0	1	0	0.0158449116	-0.009494245	-0.020898314	0.01879817	-0.008575444	-0.004841945	-0.010958642	0.01371352	0.01469286	0.02034625
4	0	1	1	0.0002980478	-0.009060399	-0.021167640	0.01976699	-0.008669477	-0.004053388	-0.010490566	0.01344312	0.01400132	0.02047635
5	0	3	0	0.0016254622	-0.009127677	-0.008551879	0.01962796	-0.008208408	-0.004546545	-0.010288477	0.01323325	0.01355036	0.01988715
6	0	3	1	0.0072630812	-0.010953918	-0.008483087	0.02027259	-0.007484172	-0.003991703	-0.008475147	0.01249293	0.01380752	0.02075384
7	1	0	0	0.0103544183	-0.009409442	-0.021051053	0.01903990	-0.008648679	-0.004712717	-0.011038853	0.01367091	0.01439632	0.02024350
8	1	0	1	0.0044314651	-0.009241268	-0.021004074	0.01958009	-0.008628553	-0.004126853	-0.010426783	0.01353494	0.01427426	0.02058184
9	1	1	0	0.0125788127	-0.006280339	-0.018763483	0.02037804	-0.007427800	-0.003929927	-0.010547775	0.01435935	0.01481264	0.02043372
10	1	1	1	0.0077508205	-0.008699636	-0.020844647	0.01943178	-0.008693791	-0.004296271	-0.011024910	0.01420559	0.01469810	0.02048203
11	1	2	0	0.0006729399	0.004996916	-0.012690374	0.02543645	-0.004623293	-0.001337389	-0.008813224	0.01430292	0.01389760	0.02057476
12	1	3	0	0.0078820959	-0.006446192	-0.003464721	0.02808316	-0.003028435	-0.002260542	-0.011520784	0.01820912	0.01609890	0.02033599
13	2	1	0	0.0084220986	-0.008630448	-0.020913445	0.01942661	-0.008394133	-0.004644940	-0.010700040	0.01339005	0.01394842	0.02025301
14	3	0	1	0.0056788353	-0.016897149	-0.005718550	0.02001873	-0.007907692	-0.004935122	-0.010333577	0.01398581	0.01453523	0.01998268

(b) Draw inference on historic volatilities.

Solution We use the same approach as with GARCH/TGARCH specifications in Question 1.

```
title_garchm <- rep(NA, times = nmods)
for (i in 1:nmods)
{
  title_garchm[i] <- paste("ARMA(",
        as.character(adq_set_arma_garchm[i, 1]), ",",
        as.character(adq_set_arma_garchm[i, 2]),
        c(")-GARCH(1,1)", ")-GARCHM(1,1)")
                                [1 + adq_set_arma_garchm[i,3]],
        sep = "")
  plot(date[-1], sqrt(
        ARMA_GARCHM_est[[adq_idx_arma_garchm[i]]]@fit$var),
        type = "l", xlab = "", ylab = "volatilities",
        ylim = c(0, 0.08), main = title_garchm[i])
}
```



All volatility estimates generally look very similar. Including the GARCH-in-mean term appears to produce only a minor difference in estimated volatilities for fixed p, q. The periods of high volatility match up to those estimated using GARCH and TGARCH models.

(c) Draw inference on the existence of a time-varying risk premium.

Solution To test for time-varying risk premia, we focus on on the coefficient of the GARCH-in-mean term in models where the GARCH-in-mean is actually included. This is δ in Engle, *et al.* (1987), but `rugarch` refers to it as "archm''.

```
for (i in 1:nmods)
{
  if (adq_set_arma_garchm[i, 3] == 1)
  {
    archm_est <- ARMA_GARCHM_est[[
                adq_idx_arma_garchm[i]]]@fit$coef["archm"]
    archm_tvl <- ARMA_GARCHM_est[[
                adq_idx_arma_garchm[i]]]@fit$tval["archm"]
    cat(paste0(title_garchm[i], ": archm_hat = ",
                    round(archm_est, 2),
                ", t-value = ",
                    round(archm_tvl, 2)),
                "\n")
  }
}
```

```
## ARMA(0,0)-GARCHM(1,1): archm_hat = 0.05, t-value = 4142.54
## ARMA(0,1)-GARCHM(1,1): archm_hat = 0.05, t-value = 3595.14
## ARMA(0,3)-GARCHM(1,1): archm_hat = 0.06, t-value = 20396.72
## ARMA(1,0)-GARCHM(1,1): archm_hat = 0.06, t-value = 117048.55
## ARMA(1,1)-GARCHM(1,1): archm_hat = 0.05, t-value = 4.83
## ARMA(3,0)-GARCHM(1,1): archm_hat = 0.05, t-value = 604.75
```

The null $H_0 : \delta = 0$ is easily rejected in favour of $H_1 : \delta > 0$ at very low significance levels for all models with a GARCH-in-mean term. Hence, we conclude that there is substantial evidence of a "time-varying risk premium".

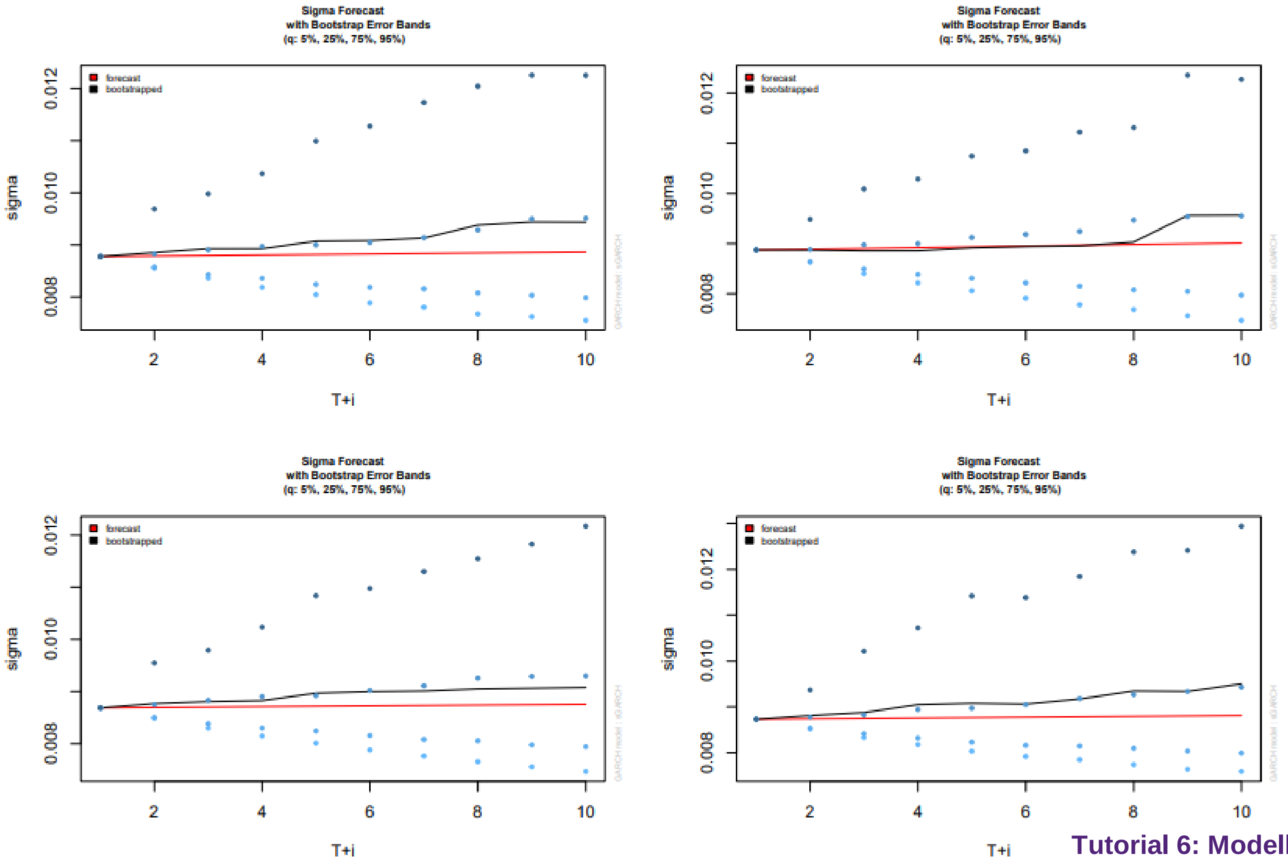
$$y_t = \mu_t + \varepsilon_t,$$

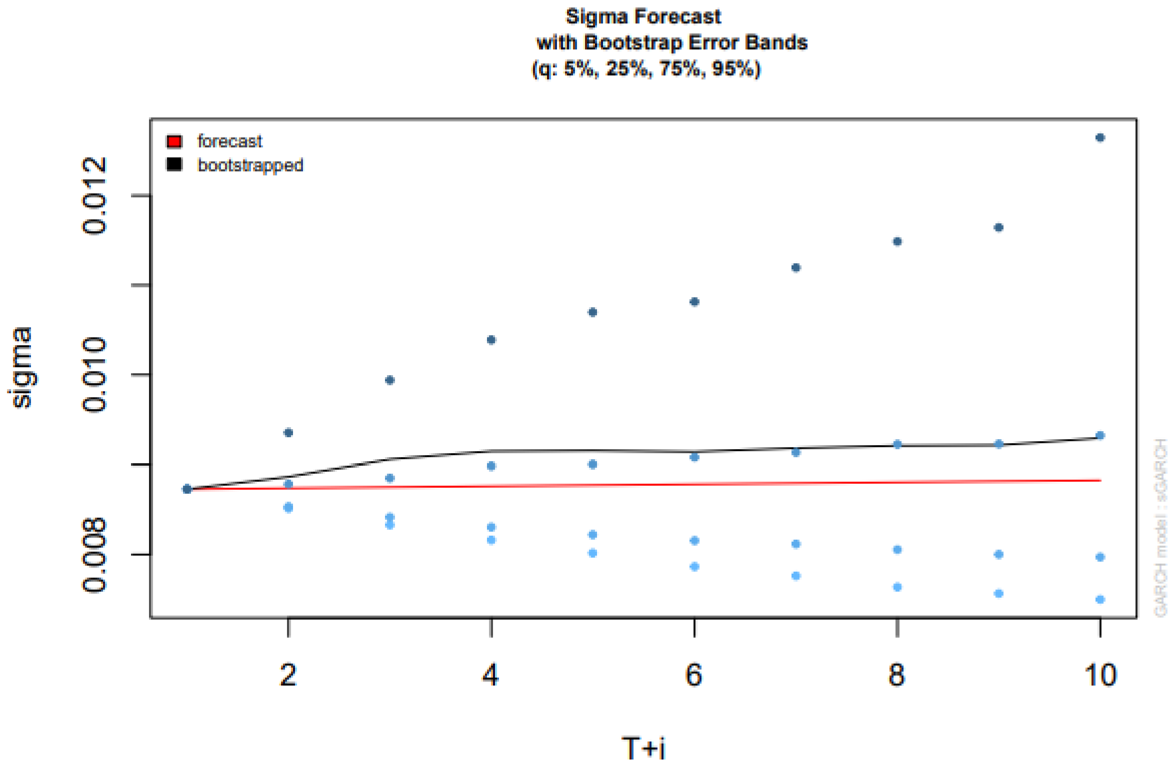
$$\mu_t = \beta + \delta\sqrt{h_t} = \beta + \delta\sqrt{\alpha_0 + \alpha_1 \sum_{j=1}^{q} w_j \varepsilon_{t-j}^2},$$

(d) Forecast volatility for the four trading days past the end of the sample.

Solution To forecast volatilities, we again use the `ugarchboot` function.

```
for (i in 1:nmods)
{
  plot(ugarchboot(ARMA_GARCHM_est[[adq_idx_arma_garchm[i]]],
                  method = "Partial"), which = 3)
}
```





The volatility forecasts obtained from GARCH-in-mean specifications are very similar to those obtained with the GARCH and TGARCH specifications.

Tutorial 6: Modelling Volatility - II

At the end of this tutorial you should be able to:

- construct an adequate set of models with possible TGARCH errors;
- construct an adequate set of models with possible GARCH-in-mean components;
- use R to draw inference on the presence of leverage effects from a class of TGARCH models;
- use R to draw inference on the presence of time-varying risk premia from a class of GARCH-in-mean models;
- use R to estimate and forecast volatilities based on models with TGARCH errors or GARCH-in-mean components.

THE UNIVERSITY OF QUEENSLAND AUSTRALIA | CREATE CHANGE

Thank you

Francisco Tavares Garcia
Academic Tutor | School of Economics
tavaresgarcia.github.io

Reference

Tsay, R. (2010). Analysis of Financial Time Series, 3rd Edition, John Wiley & Sons.