Modelo ARCH(m).

Teoría.

Dr. Martín Lozano `https://mlozanoqf.github.io/`

30 de octubre de 2025, 03:07 a.m.

	Fundamental	Intermedio	Especializado
Finanzas	×	✓	×
Estadística	×	✓	×
R	×	×	×

1 Introducción.

En negocios necesitamos datos para evaluar y fundamentar decisiones.

- **Partimos de un propósito.** Un problema que resolver, una hipótesis que validar, un objetivo que alcanzar o una tarea que ejecutar.
- **Datos.** Sin información, nuestras conclusiones serían simples opiniones.
- **Analizamos y modelamos.** Usamos los datos para analizar, estimar o entrenar modelos que permitan un análisis empírico riguroso.
- **Generamos evidencia.** Los resultados nos ayudan a validar hipótesis, resolver problemas y generar nuevo conocimiento para tomar decisiones.

2 Paquetes.

```
library(tidyquant)
library(dplyr)
library(knitr)
library(ggplot2)
library(tidyr)
library(scales)
```

3 Descarga de datos.

```
S <- tq_get("^GSPC",
                      from = "2015-07-10",
                      to   = "2020-07-10") %>%
  dplyr::select(date, S = close)



kable(
  rbind(head(S, 6), tail(S, 2)),
  digits = 10,
  format.args = list(scientific = FALSE)
)
```

date	S
2015-07-10	2076.62
2015-07-13	2099.60
2015-07-14	2108.95
2015-07-15	2107.40
2015-07-16	2124.29
2015-07-17	2126.64
2020-07-08	3169.94
2020-07-09	3152.05

4 Rendimientos logaritmicos.

$$u_i = \ln\left(\frac{S_i}{S_{i-1}}\right)$$

$u_2 = \ln\left(\frac{2099.60}{2076.62}\right) \to \ln(1.011066) = 0.01100527$

```
S <- S %>%
  mutate(u_log = log(S / lag(S)))

kable(
  rbind(head(S, 6), tail(S, 2)),
  digits = 10,
  format.args = list(scientific = FALSE)
)
```

date	S	u_log
2015-07-10	2076.62	NA
2015-07-13	2099.60	0.0110052685
2015-07-14	2108.95	0.0044432732
2015-07-15	2107.40	-0.0007352563
2015-07-16	2124.29	0.0079827334
2015-07-17	2126.64	0.0011055716
2020-07-08	3169.94	0.0077969863
2020-07-09	3152.05	-0.0056595915

$$\sigma_n^2 = \frac{1}{m-1}\sum_{i=1}^{m}(u_{n-i} - \bar{u})^2$$

$$(u_{n-i} - \bar{u})^2$$

```r
u_mean <- mean(S$u_log, na.rm = TRUE)

S <- S %>%
  mutate(u_dev2 = (u_log - u_mean)^2)

kable(
  rbind(head(S, 6), tail(S, 2)),
  digits = 10,
  format.args = list(scientific = FALSE)
)
```

date	S	u_log	u_dev2
2015-07-10	2076.62	NA	NA
2015-07-13	2099.60	0.0110052685	0.0001139245
2015-07-14	2108.95	0.0044432732	0.0000169048
2015-07-15	2107.40	-0.0007352563	0.0000011385
2015-07-16	2124.29	0.0079827334	0.0000585379
2015-07-17	2126.64	0.0011055716	0.0000005988
2020-07-08	3169.94	0.0077969863	0.0000557301
2020-07-09	3152.05	-0.0056595915	0.0000358959

$$\sigma_n^2 = \frac{1}{m-1}\sum_{i=1}^{m}(u_{n-i} - \bar{u})^2$$

$$\sigma_n = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(u_{n-i} - \bar{u})^2}$$

```r
n_valid <- sum(!is.na(S$u_dev2))
s2_n <- sum(S$u_dev2, na.rm = TRUE) / (n_valid - 1)
s2_n
```

```
## [1] 0.0001505991
```

```
s2_n^.5
```

```
## [1] 0.01227188
```

$$\sigma_n^2 = 0.0001505991$$
$$\sigma_n = 0.01227188$$

5 Rendimientos como cambios porcentuales.

$$u_i = \frac{S_i - S_{i-1}}{S_{i-1}}$$

$u_2 = \frac{2099.60-2076.62}{2076.62} \rightarrow \frac{22.98}{2076.62} = 0.011066$

```
S <- tq_get("^GSPC",
                     from = "2015-07-10",
                     to   = "2020-07-10") %>%
  dplyr::select(date, S = close)

S <- S %>%
  mutate(u_pc = (S / lag(S))-1)

kable(
  rbind(head(S, 6), tail(S, 2)),
  digits = 10,
  format.args = list(scientific = FALSE)
)
```

date	S	u_pc
2015-07-10	2076.62	NA
2015-07-13	2099.60	0.0110660492
2015-07-14	2108.95	0.0044531592
2015-07-15	2107.40	-0.0007349861
2015-07-16	2124.29	0.0080146804
2015-07-17	2126.64	0.0011061830
2020-07-08	3169.94	0.0078274619
2020-07-09	3152.05	-0.0056436062

6 $\bar{u}$ is assumed to be zero.

```
S <- S %>%
  mutate(u_pc2 = dplyr::lag(u_pc^2))

kable(
  rbind(head(S, 6), tail(S, 2)),
  digits = 10,
  format.args = list(scientific = FALSE)
)
```

date	S	u_pc	u_pc2
2015-07-10	2076.62	NA	NA
2015-07-13	2099.60	0.0110660492	NA
2015-07-14	2108.95	0.0044531592	0.0001224574
2015-07-15	2107.40	-0.0007349861	0.0000198306
2015-07-16	2124.29	0.0080146804	0.0000005402
2015-07-17	2126.64	0.0011061830	0.0000642351
2020-07-08	3169.94	0.0078274619	0.0001170406
2020-07-09	3152.05	-0.0056436062	0.0000612692

$$\sigma_n^2 = \frac{1}{m}\sum_{i=1}^{m} u_{n-i}^2$$

$$\sigma_n = \sqrt{\frac{1}{m}\sum_{i=1}^{m} u_{n-i}^2}$$

```
n_valid <- sum(!is.na(S$u_pc2))
s2_n <- sum(S$u_pc2, na.rm = TRUE) / (n_valid - 1)
s2_n
```

```
## [1] 0.0001492955
```

```
s2_n^.5
```

```
## [1] 0.01221865
```

$$\sigma_n^2 = 0.0001492955$$

$$\sigma_n = 0.01221865$$

```
# Parámetros del modelo
omega <- 0.0000039818
alpha <- 0.223793
beta  <- 0.747577

# Inicializar u_pc2 desplazado una observación hacia abajo
S <- S %>%
  mutate(u_pc2 = lag(u_pc^2))

# Calcular recursivamente a partir del segundo valor no NA
for (i in 2:nrow(S)) {
  if (!is.na(S$u_pc[i - 1]) && !is.na(S$u_pc2[i - 1])) {
    S$u_pc2[i] <- omega +
                  alpha * (S$u_pc[i - 1]^2) +
                  beta  *  S$u_pc2[i - 1]
  }
}

kable(
  rbind(head(S, 6), tail(S, 2)),
  digits = 10,
  format.args = list(scientific = FALSE)
)
```

date	S	u_pc	u_pc2
2015-07-10	2076.62	NA	NA
2015-07-13	2099.60	0.0110660492	NA
2015-07-14	2108.95	0.0044531592	0.0001224574
2015-07-15	2107.40	-0.0007349861	0.0000999661
2015-07-16	2124.29	0.0080146804	0.0000788351
2015-07-17	2126.64	0.0011061830	0.0000772925
2020-07-08	3169.94	0.0078274619	0.0001672905
2020-07-09	3152.05	-0.0056436062	0.0001427560

```r
n_valid <- sum(!is.na(S$u_pc2))
n_valid
```

```
## [1] 1257
```

```r
s2_n <- sum(S$u_pc2, na.rm = TRUE) / (n_valid)
s2_n
```

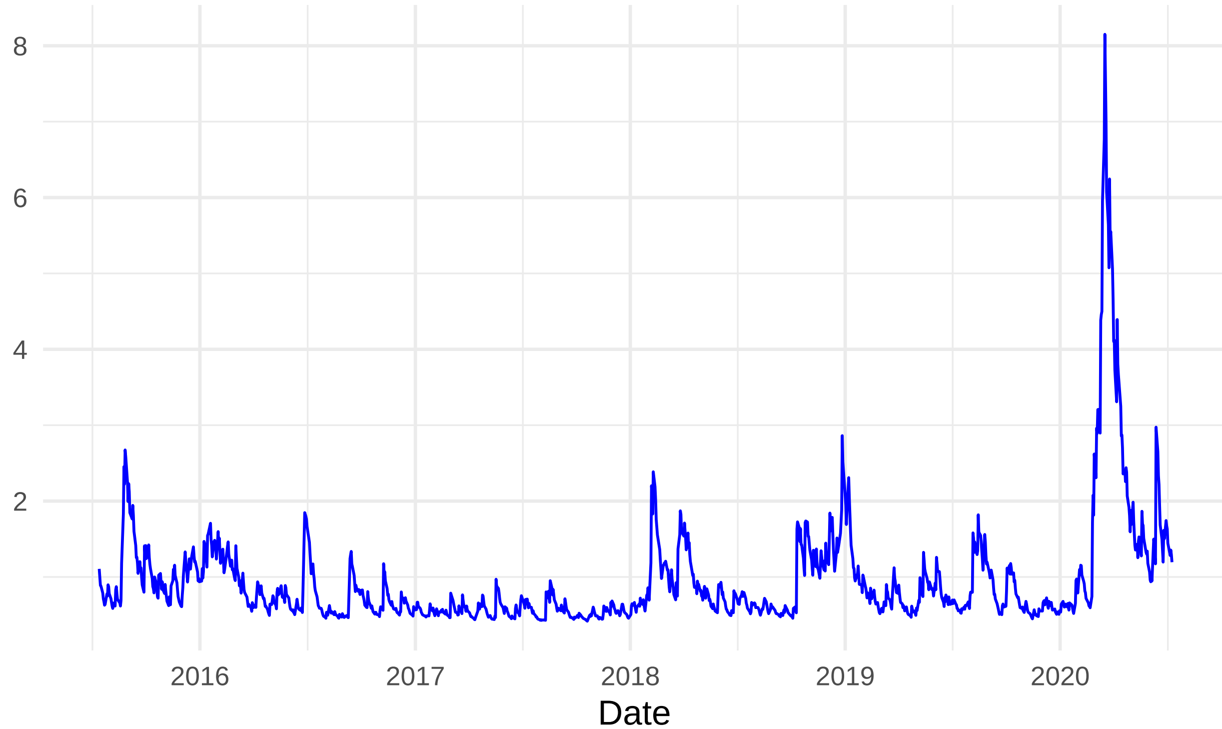
```
## [1] 0.000147982
```

```r
s2_n^.5
```

```
## [1] 0.01216479
```

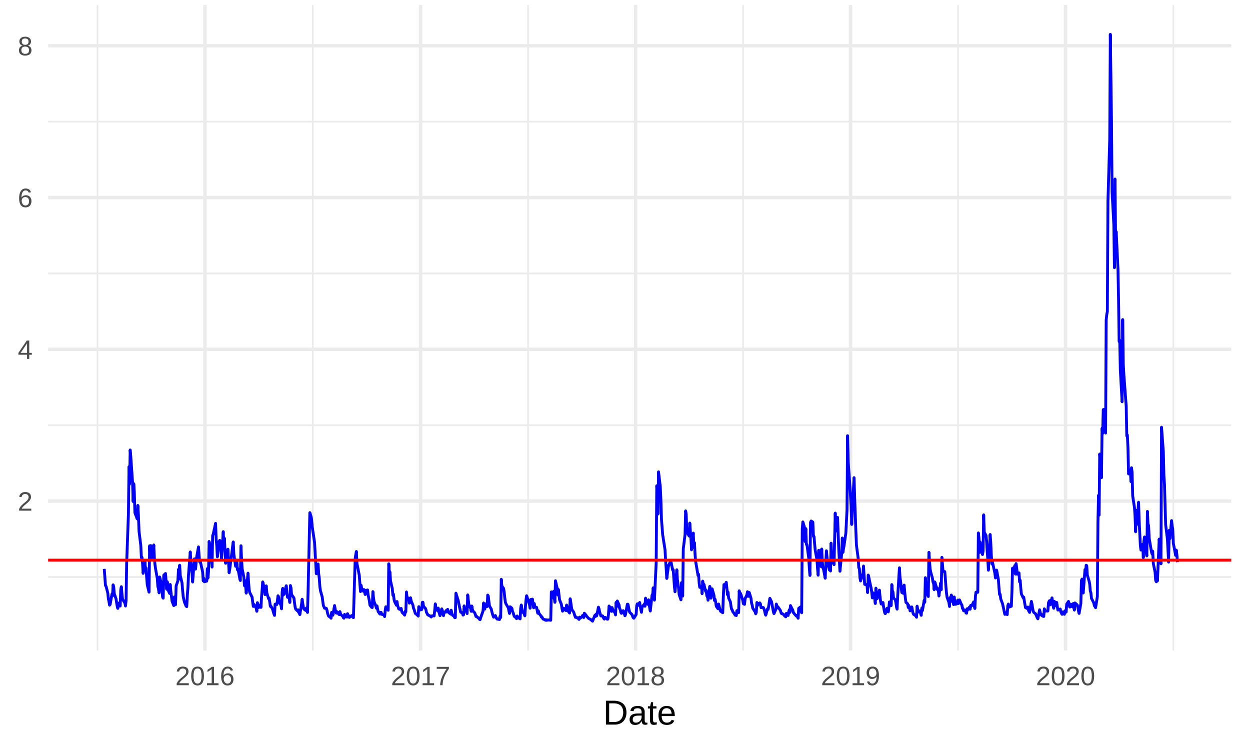
```r
# Asegúrate de tener u_pc2 calculado antes de esto

# Gráfico de la serie u_pc2
ggplot(S, aes(x = date, y = 100*u_pc2^.5)) +
  geom_line(color = "blue") +
  labs(
    title = "Figure 23.2 Volatility (% per day) of S&P 500 index:
    July 10, 2015, to July 9, 2020.",
    y = NULL,
    x = "Date") +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.title.y = element_text(angle = 0, vjust = 0.5)
  )
```

**Figure 23.2 Volatility (% per day) of S&P 500 index:
July 10, 2015, to July 9, 2020.**




```r
ggplot(S, aes(x = date, y = 100*u_pc2^.5)) +
  geom_line(color = "blue") +
    geom_hline(yintercept = 1.221865, color = "red") +
  labs(
    title = "Figure 23.2 Volatility (% per day) of S&P 500 index:
    July 10, 2015, to July 9, 2020.",
    y = NULL,
    x = "Date") +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.title.y = element_text(angle = 0, vjust = 0.5)
  )
```

Figure 23.2 Volatility (% per day) of S&P 500 index: July 10, 2015, to July 9, 2020.




```r
garch_variance_shifted <- function(u, omega, alpha, beta) {
  n <- length(u)
  v <- rep(NA_real_, n)    # v[1] = NA por convención
  if (n >= 2) v[2] <- u[1]^2
  if (n >= 3) {
    for (i in 3:n) {
      v[i] <- omega + alpha * u[i - 1]^2 + beta * v[i - 1]
    }
  }
  v
}
```

```r
# u_i: rendimientos en S$u_pc
u <- S$u_pc
u <- u[!is.na(u)]              # quita NAs (p.ej., por lag en su construcción)

# Neg-LogLik de (23.12) (para minimizar)
nll_garch_shifted <- function(par, u) {
  omega <- par[1]; alpha <- par[2]; beta <- par[3]
  # restricciones básicas
  if (omega <= 0 || alpha < 0 || beta < 0 || (alpha + beta) >= 1) return(1e12)
  v <- garch_variance_shifted(u, omega, alpha, beta)
  mask <- !is.na(v) & v > 0
  if (!any(mask)) return(1e12)
  -sum( -log(v[mask]) - (u[mask]^2) / v[mask] )
}

start <- c(omega = 1e-6, alpha = 0.1, beta = 0.8)
```

```

fit <- optim(
  par    = start,
  fn     = nll_garch_shifted,
  u      = u,
  method = "L-BFGS-B",
  lower  = c(1e-12, 0, 0),
  upper  = c(Inf, 1 - 1e-6, 1 - 1e-6)
)

theta_hat <- fit$par
omega_hat <- theta_hat[1]; alpha_hat <- theta_hat[2]; beta_hat <- theta_hat[3]

# Reconstrucción de v_i y evaluación de (23.12)
v_hat  <- garch_variance_shifted(u, omega_hat, alpha_hat, beta_hat)
mask   <- !is.na(v_hat) & v_hat > 0                 # debería ser i >= 2
ell_i  <- -log(v_hat[mask]) - (u[mask]^2) / v_hat[mask]
ll_tot <- sum(ell_i)                                # -fit$value

# Anexar a S (alineado con el índice de u en S)
idx <- which(!is.na(S$u_pc))
S$vi_mle_shifted    <- NA_real_
S$ell_23_12_shifted <- NA_real_
S$vi_mle_shifted[idx]          <- v_hat
S$ell_23_12_shifted[idx[mask]]  <- ell_i   # solo donde hay v válido

# Métricas
n_obs <- sum(mask)                # debería ser 1257 si u tiene 1258 obs
k     <- 3L
AIC   <- -2 * ll_tot + 2 * k
BIC   <- -2 * ll_tot + log(n_obs) * k

list(par = theta_hat, logLik = ll_tot, n_obs = n_obs, AIC = AIC, BIC = BIC)
```

```
## $par
##        omega        alpha         beta
## 9.116281e-07 1.795347e-01 8.176743e-01
##
## $logLik
## [1] 10781.9
##
## $n_obs
## [1] 1257
##
## $AIC
## [1] -21557.8
##
## $BIC
## [1] -21542.39
```

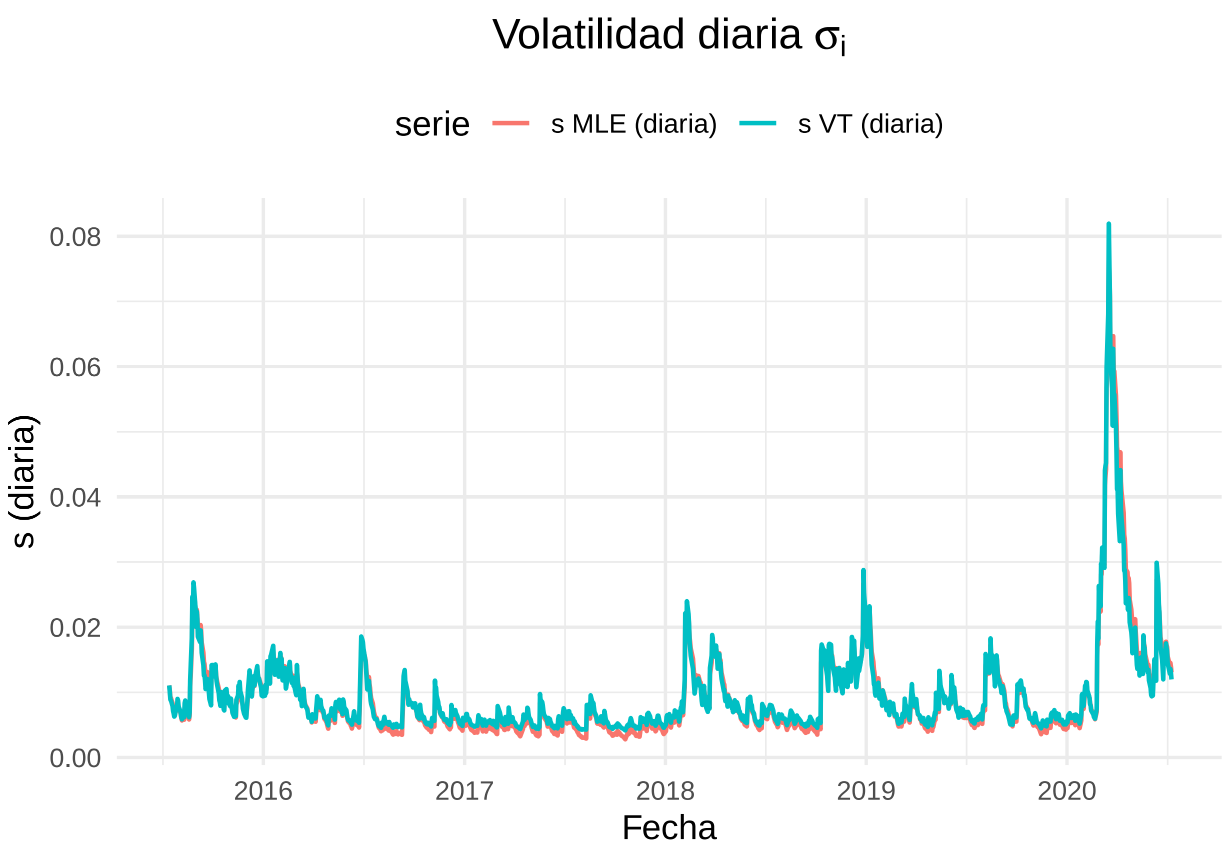
```
# Varianza muestral de u para variance targeting
VL <- var(u, na.rm = TRUE)

nll_garch_VT_shifted <- function(par, u, VL) {
  alpha <- par[1]; beta <- par[2]
  if (alpha < 0 || beta < 0 || (alpha + beta) >= 1) return(1e12)
```

```r
  omega <- VL * (1 - alpha - beta)
  v <- garch_variance_shifted(u, omega, alpha, beta)
  mask <- !is.na(v) & v > 0
  if (!any(mask)) return(1e12)
  -sum( -log(v[mask]) - (u[mask]^2) / v[mask] )
}

fit_vt <- optim(
  par    = c(alpha = 0.2, beta = 0.7),
  fn     = nll_garch_VT_shifted,
  u      = u,
  VL     = VL,
  method = "L-BFGS-B",
  lower  = c(0, 0),
  upper  = c(1 - 1e-6, 1 - 1e-6)
)

alpha_vt <- fit_vt$par[1]
beta_vt  <- fit_vt$par[2]
omega_vt <- VL * (1 - alpha_vt - beta_vt)

# Evaluación final de (23.12) con VT
v_vt   <- garch_variance_shifted(u, omega_vt, alpha_vt, beta_vt)
mask_vt <- !is.na(v_vt) & v_vt > 0
ell_vt <- -log(v_vt[mask_vt]) - (u[mask_vt]^2) / v_vt[mask_vt]
ll_vt  <- sum(ell_vt)

n_obs_vt <- sum(mask_vt)         #   1257
k_vt     <- 2L
AIC_vt   <- -2 * ll_vt + 2 * k_vt
BIC_vt   <- -2 * ll_vt + log(n_obs_vt) * k_vt

list(par = c(omega = omega_vt, alpha = alpha_vt, beta = beta_vt),
     logLik = ll_vt, n_obs = n_obs_vt, AIC = AIC_vt, BIC = BIC_vt)
```

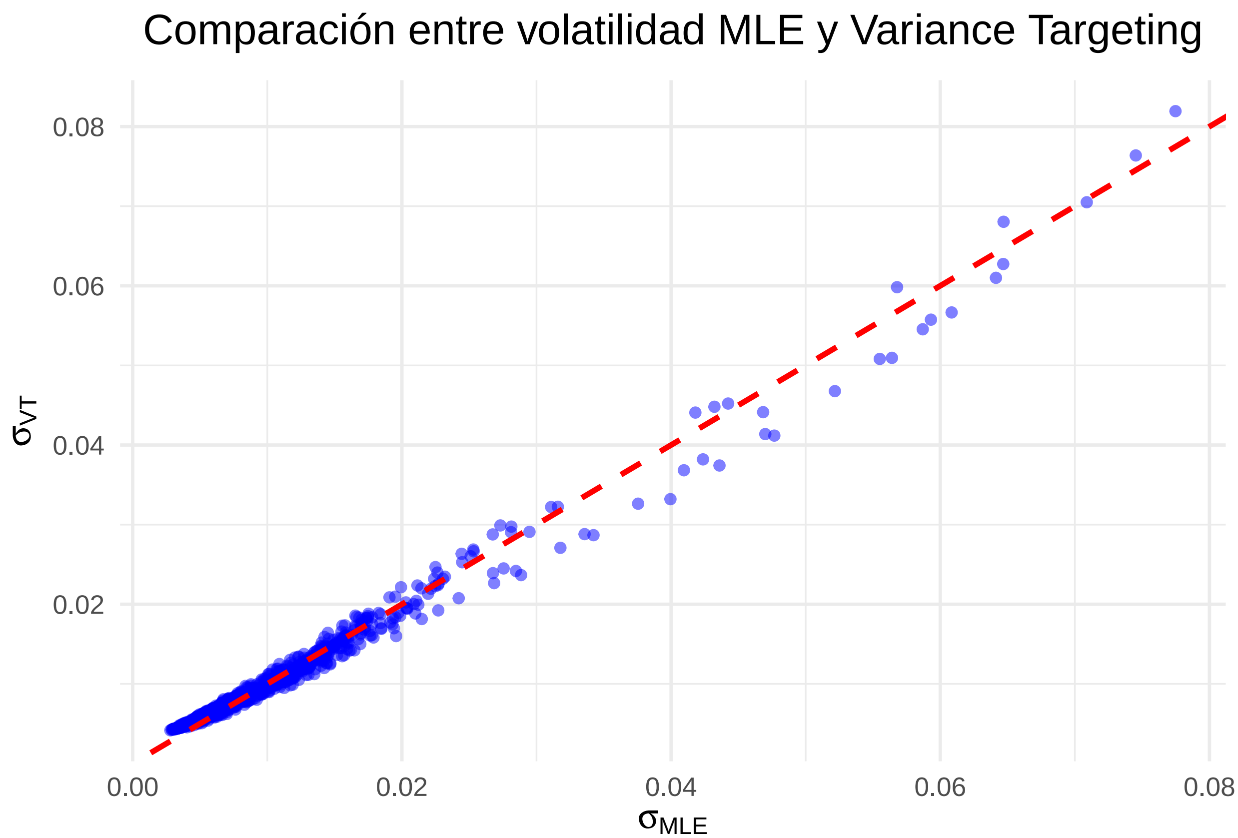
```
## $par
##  omega.alpha  alpha.alpha    beta.beta
## 3.966362e-06 2.263490e-01 7.470377e-01
##
## $logLik
## [1] 10837.4
##
## $n_obs
## [1] 1257
##
## $AIC
## [1] -21670.81
##
## $BIC
## [1] -21660.54
```

```r
# Asumo que ya tienes:
# - v_hat  : trayectoria v_i por MLE (shifted: v2 = u1^2)
# - v_vt   : trayectoria v_i por Variance Targeting (shifted)
# - S$date : fechas
# - S$u_pc : rendimientos
```

```r

# 1) Volatilidad diaria: sigma_i = sqrt(v_i)
sigma_mle_daily <- sqrt(v_hat)
sigma_vt_daily  <- sqrt(v_vt)

# 2) (OPCIONAL) Volatilidad anualizada: sigma_annual = sqrt(252 * v_i)
sigma_mle_ann <- sqrt(252 * v_hat)
sigma_vt_ann  <- sqrt(252 * v_vt)

# Alinear con S (suponiendo idx = posiciones donde S$u_pc no es NA)
idx <- which(!is.na(S$u_pc))

# Guarda en S para inspección/uso posterior
S$sigma_mle_daily_shifted <- NA_real_
S$sigma_vt_daily_shifted  <- NA_real_
S$sigma_mle_ann_shifted   <- NA_real_
S$sigma_vt_ann_shifted    <- NA_real_

S$sigma_mle_daily_shifted[idx] <- sigma_mle_daily
S$sigma_vt_daily_shifted[idx]  <- sigma_vt_daily
S$sigma_mle_ann_shifted[idx]   <- sigma_mle_ann
S$sigma_vt_ann_shifted[idx]    <- sigma_vt_ann

# ======= Elige qué graficar =======
# Opción A) Volatilidad DIARIA (en proporción, o multiplica por 100 si prefieres %)
df_plot_daily <- S %>%
  select(date,
         ` MLE (diaria)` = sigma_mle_daily_shifted,
         ` VT (diaria)`  = sigma_vt_daily_shifted) %>%
  pivot_longer(-date, names_to = "serie", values_to = "valor")

ggplot(df_plot_daily, aes(x = date, y = valor, color = serie)) +
  geom_line(na.rm = TRUE, linewidth = 0.8) +
  labs(title = expression("Volatilidad diaria " * sigma[i]),
       x = "Fecha", y = "  (diaria)") +
  theme_minimal(base_size = 13) +
  theme(plot.title = element_text(face = "bold", hjust = 0.5),
        legend.position = "top")
```




```r
# Asegúrate de tener:
# sigma_mle_daily, sigma_vt_daily  (o las anualizadas si prefieres)
df_scatter <- data.frame(
  sigma_MLE = sigma_mle_daily,
  sigma_VT  = sigma_vt_daily
)

# Quitar NAs
df_scatter <- na.omit(df_scatter)

# Gráfico
ggplot(df_scatter, aes(x = sigma_MLE, y = sigma_VT)) +
  geom_point(alpha = 0.5, color = "blue") +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed", linewidth = 1) +
  labs(
    title = expression("Comparación entre volatilidad MLE y Variance Targeting"),
    x = expression(sigma[MLE]),
    y = expression(sigma[VT])
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5)
  )
```



Comparación de estimación MLE vs Variance Targeting

La ecuación (23.12) del modelo GARCH(1,1) representa la **función de log-verosimilitud** a maximizar:

$$\ln L = -\frac{1}{2} \sum_{i=1}^{n} \left[\ln(v_i) + \frac{u_i^2}{v_i} \right],$$

donde v_i es la varianza condicional obtenida recursivamente como

$$v_i = \omega + \alpha u_{i-1}^2 + \beta v_{i-1}.$$

6.0.1 Evaluación de los modelos

Modelo	Parámetros estimados	Log-verosimilitud	AIC	BIC
MLE	ω, α, β	menor	mayor	mayor
Variance Targeting	α, β	**mayor**	**menor**	**menor**

6.0.2 Interpretación

El modelo de **Variance Targeting (VT)** alcanza un valor más alto de la función de log-verosimilitud, lo que indica un **mejor ajuste** a los datos observados u_i. Además, sus valores más bajos de **AIC** y **BIC** muestran que logra dicho ajuste con **menor complejidad** (menos parámetros).

Esto es coherente con las fórmulas de penalización:

$$\text{AIC} = -2\ln(\hat{L}) + 2k, \qquad \text{BIC} = -2\ln(\hat{L}) + k\ln(n),$$

donde k es el número de parámetros y n el tamaño muestral.

6.0.3 Conclusión

> Dado que el modelo **Variance Targeting** presenta **mayor log-verosimilitud**
> y **menores valores de AIC y BIC**,
> se concluye que ofrece el **mejor equilibrio entre ajuste y parsimonia**.
>
> Por tanto, se selecciona el **GARCH(1,1) con variance targeting**
> como el modelo preferido para describir la dinámica de volatilidad.

7 Conclusión.

En negocios necesitamos datos para evaluar y fundamentar decisiones.

- R facilita el análisis financiero reproducible al integrar en un solo flujo la descarga, transformación y visualización de datos con paquetes como `tidyquant`, `dplyr` y `ggplot2`.
- Este enfoque promueve decisiones financieras mejor fundamentadas.