# How to Implement a Volatility Model Using R?

# ¿Cómo Implementar un Modelo de Volatilidad usando Lenguaje R?

**Fernán Villa; Juan David Velásquez; Paola A. Sánchez**
*Grupo Computación Aplicada - Universidad Nacional de Colombia*
fernanvilla@gmail.com; jdvelasq@unal.edu.co; pasanche@unal.edu.co

(Artículo de INVESTIGACIÓN CIENTÍFICA Y TECNOLÓGICA. Recibido el 07/08/2011. Aprobado el 10/11/2011)

***Abstract***
Modeling and forecasting of financial time series is an activity of economic interest for agent's market. Time series from this area often displays complex dynamic relationships between its variables and it can be captured by volatility models. These models can be implemented in most existing programming environments. However, the license of these environments is not free; in addition, the implementation is complex because there aren't guidelines for designing and implementing of the code. R is a free and stable programming environment, so this paper suggests some guidelines for designing and implementing the code in R of an volatility model; like an example case, this paper proposes the creation of Volatility package that includes the GARCH model and show its applicability in an volatility model of a real time series.

***Keywords***: Volatility, Model, Implementation, GARCH, R Languages

**Resumen**
El modelado y pronóstico de series de tiempo financieras es una actividad de interés económico para los agentes del mercado. Las series de tiempo provenientes de ésta área a menudo presentan relaciones dinámicas complejas entre sus variables las cuales pueden ser capturadas mediante modelos de volatilidad. Estos pueden ser implementados en la mayoría de entornos de programación existentes. Sin embargo, la implementación de los modelos es compleja por no tener pautas para diseñar e implementar su código. Dado que R es un entorno de programación gratuito y estable, en este trabajo se proponen algunas pautas para diseñar e implementar el código de un modelo de volatilidad en R; como caso de ejemplo, se propone la creación del paquete Volatility que incluye el modelo de volatilidad GARCH y se mostrará su aplicabilidad al modelar la volatilidad de una serie de tiempo real.

**Palabras clave:** Volatilidad, Modelo, Implementación, GARCH, Lenguaje R

## I. INTRODUCCIÓN

Las series de tiempo financieras, a menudo, presentan relaciones dinámicas complejas entre sus variables, las cuales pueden ser capturadas mediante modelos de volatilidad. Estos pueden implementarse en la mayoría de lenguajes de programación existentes, entre otros: R, MATLAB, SAS, *Mathematica, Visual Studio .Net* y *Java*. Sin embargo, la licencia de la mayoría de los lenguajes mencionados no es gratuita, lo que implica costos adicionales para quien necesita realizar una implementación de algún modelo, ó simplemente desee brindar algún tipo de consultoría; además, sin importar el lenguaje, la implementación de un modelo es compleja por no tener pautas para diseñar e implementar su código.

Actualmente, existen lenguajes de programación de licencia gratuita ó GNU (*General Public License*) lo que permite realizar implementaciones de software de manera gratuita, entre estos se encuentra R (*R Development Core Team* 2008); éste lenguaje es utilizado en investigación por la comunidad estadística, siendo además popular en otros campos de investigación y aplicación, como la bioinformática y las matemáticas financieras.

R es un lenguaje de programación robusto y estable, en el cual es posible crear y cargar diferentes librerías o paquetes con finalidades específicas de cálculo; de modo que, un paquete puede contener la implementación de una o varias funciones, y por tanto de uno o varios modelos. La programación organizada en paquetes facilita

la distribución y utilización del (los) modelo(s) implementado(s).

Entonces, con el fin de aprovechar las ventajas mencionadas y de facilitar la implementación de un modelo de volatilidad en R, en éste trabajo se proponen algunas pautas para el diseño y desarrollo de un modelo GARCH [1]. Estas pueden ser seguidas para realizar la implementación de otro tipo de modelo de volatilidad.

Para cumplir con el objetivo propuesto éste trabajo se encuentra organizado como sigue: en la sección 2 se definen las ecuaciones del modelo de volatilidad GARCH; en la 3, se realiza la implementación de un modelo GARCH(1,1) en *Microsoft Excel* para modelar la volatilidad de la TRM Colombiana, con el fin de entender las ecuaciones del modelo GARCH; en la 4, se definen algunas pautas de diseño y se realiza la implementación del modelo GARCH en R; en la 5, se utiliza el modelo implementado para modelar la volatilidad TRM Colombiano y comprobar su funcionalidad; finalmente, se concluye.

## II. MODELO DE VOLATILIDAD GARCH

El modelado y pronóstico de series de tiempo financieras es una actividad de interés económico para los agentes del mercado. Las series de tiempo provenientes de ésta área a menudo presentan relaciones dinámicas complejas entre sus variables las cuales pueden ser capturadas mediante modelos de volatilidad.

En [3] se desarrolla el modelo ARCH (Autoregresivo Condicionalmente Heterocedástico), el cual permite modelar la dinámica de la volatilidad de una serie temporal. Años después, [1] propone GARCH (Generalizado Autoregresivo Condicionalmente Heterocedástico), siendo este una ampliación del modelo ARCH, en el que se incluyen rezagos de la varianza condicional. Para más detalles sobre el modelo GARCH consultar [2].

En el modelo *GARCH(p,q)*, la volatilidad está dada por [1]:

$$\sigma_t^2 = \delta_0 + \left[\sum_{i=1}^{p} \delta_i e_{t-i}^2\right] + \left[\sum_{i=1}^{q} \beta_i \sigma_{t-i}^2\right] \quad (1)$$

Donde, *p* es la cantidad de rezagos de la serie, y *q* la cantidad de rezagos de la volatilidad o varianza condicional, *e* puede ser calculado mediante:

$$e_t = y_t - \hat{y}_t \quad (2)$$

Para la ecuación (2), $\hat{y}_t$ corresponde a un modelo autoregresivo *AR(p)*, el cual especifica que una serie estacionaria en un tiempo *t* depende de *p* valores pasados de la misma (rezagos), ponderados por un peso $\alpha_i$ que mide la influencia del rezago en el valor presente de la serie, el modelo esta dado por:

$$\hat{y}_t = \alpha_0 + \left[\sum_{i=1}^{p} \alpha_i y_{t-i}\right] \quad (3)$$

Los parámetros del los modelos (1) y (3) se estiman de forma simultánea maximizando el logaritmo de la función de verosimilitud de los residuales, dada por:

$$\log L = -\frac{T}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{T}\log(\sigma_t^2) - \frac{1}{2}\sum_{t=1}^{T}\frac{e_t^2}{\sigma_t^2} \quad (4)$$

Los parámetros del modelo también se pueden estimar minimizando la ecuación dual de (4), es decir, *–log L*.

Además, de la ecuación (1) se deben satisfacer las condiciones: de no negatividad $\delta_0>0$, $\delta_i\geq 0$, $\beta_i\geq 0$, necesarias para garantizar que la varianza sea positiva en todos los periodos ($\sigma_t^2 > 0$ para todo *t*); y de estacionariedad dada por:

$$\left[\sum_{i=1}^{máx\{p,q\}} \delta_i + \beta_i\right] < 1 \quad (5)$$

Mientras que, de la ecuación (3) se tienen las condiciones: $\alpha_0>0$, $\alpha_i\geq 0$.

Consecuentemente, el orden de uso de las ecuaciones es: primero calcular (3); luego (2); teniendo $e_t$, se procede con (1); y finalmente, se estiman los parámetros maximizando (4), con las restricciones mencionadas.

## III. IMPLEMENTACIÓN DEL MODELO GARCH EN *MICROSOFT EXCEL*

Antes de comenzar la implementación del modelo GARCH en R es necesario entender a cabalidad las ecuaciones del mismo, para esto, es conveniente realizar un ejemplo en *Microsoft Excel* u otra hoja de cálculo. La implementación

en Excel permite tener un primer acercamiento a la implementación del modelo GARCH, y tener un caso de prueba para el desarrollo en R, en ésta sección se ilustra cómo realizarlo en Microsoft Excel, para modelar la volatilidad de una serie específica. Una vez se han entendido todas las ecuaciones, se puede proceder con la implementación en R.

Para la implementación se tendrán en cuenta las siguientes consideraciones: se modela la volatilidad de TRM colombiana (tasa de cambio representativa del mercado) entre el 1 de enero del 2000 y el 30 de septiembre de 2009, ver Figura 1; se modelará con *GARCH(1,1)* para facilitar la explicación de la implementación. A continuación se explicarán los pasos a seguir para realizar el desarrollo.

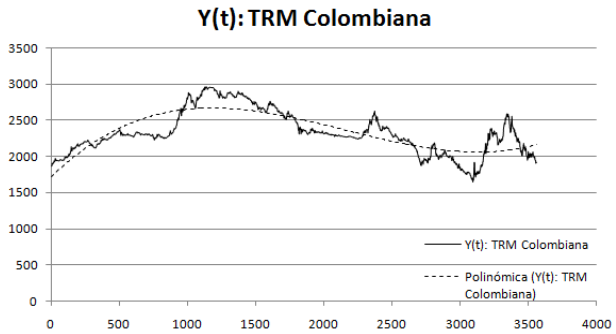


Fig. 1. $y_t$: TRM Colombiana.

**Primer Paso:** determinar las ecuaciones para el modelo *GARCH(1,1).* El modelo *AR(1)* asociado es el resultante de la ecuación (3), con *p=1*:

$$\hat{y}_t = \alpha_0 + \alpha_1 y_{t-1} \quad (6)$$

El error está dado por la ecuación (2), mientras que *GARCH(1,1)* es:

$$\sigma_t^2 = \delta_0 + \delta_1 e_{t-1}^2 + \beta_1 \sigma_{t-1}^2 (7)$$

**Segundo Paso:** especificar las celdas donde se almacenarán los valores de los parámetros del modelo que se van a estimar. En la Figura 2, los parámetros $\alpha_0$, $\alpha_1$, $\delta_0$, $\delta_1$, $\beta_1$, corresponden a las celdas adyacentes a las etiquetas *alpha0, alpha1, delta0, delta1, betha1.* Para facilitar la optimización de la ecuación (4) las celdas *M2 a M6,* se inicializan con el valor de *0,0001;* la celda adyacente a *delta1+betha1 (M7=M5+M6)* se utiliza para tener en cuenta la restricción (5), es decir, $\delta_1+\beta_1<1$.

| | K | L | M |
|---|---|---|---|
| 1 | | Parámetros | |
| 2 | | alpha0 | 0,00010 |
| 3 | | alpha1 | 0,00010 |
| 4 | | delta0 | 0,00010 |
| 5 | | delta1 | 0,00010 |
| 6 | | betha1 | 0,00010 |
| 7 | | delta1+betha1 | 0,00020 |

Fig.2. Parámetros del Modelo.

**Tercer Paso:** diferenciar la serie en caso de ser necesario. Con el fin de eliminar la tendencia presente en la serie TRM (ver Figura 1), se procede a diferenciar la serie como se puede apreciar en la columna D de la Figura 3. Entonces, $y_{(t)} = TRM_{(t)} - TRM_{(t-1)}$.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Dato | Día | TRM(t) | y(t)= TRM(t)-TRM(t-1) |
| 2 | 1 | 01/01/2000 | 1873,77 | |
| 3 | 2 | 02/01/2000 | 1873,77 | =C3-C2 |
| 4 | 3 | 03/01/2000 | 1873,77 | 0 |
| 5 | 4 | 04/01/2000 | 1874,35 | 0,58 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Fig. 3. Diferenciación de la serie TRM.

**Cuarto Paso:** calcular el respectivo modelo *AR.* Entonces, se agrega una columna para el primer rezago ($y_{t-1}$), columna E en la Figura 4; y otra para calcular $\hat{y}_t$, columna F en la Figura 5, correspondiente a la ecuación (6).

| C | D | E |
|---|---|---|
| TRM(t) | y(t)= TRM(t)-TRM(t-1) | y(t-1) rezago 1 |
| 1873,77 | | |
| 1873,77 | 0 | |
| 1873,77 | 0 | =D3 |
| 1874,35 | 0,58 | 0 |
| ⋮ | ⋮ | ⋮ |

Fig. 4. Columna E para el primer rezago $y_{t-1}$.

**Quinto Paso:** calcular el error con base en la ecuación (2), como se presenta en la Figura 6, donde se usa la columna G para calcular $e_t$.

**Sexto Paso:** calcular el modelo *GARCH(1,1)*. La ecuación (7), se calcula en la columna H de la Figura 7

| | E | F | K | L | M |
|---|---|---|---|---|---|
| 1 | Y(t-1) | Ŷ(t) | | Parámetros | |
| 2 | | | | alpha0 | 0,00010 |
| 3 | | | | alpha1 | 0,00010 |
| 4 | 0 | =$M$2+$M$3*E4 | | | 0,00010 |
| 5 | 0 | 0,0001 | | delta1 | 0,00010 |
| 6 | 0,58 | 0,000158 | | betha1 | 0,00010 |
| 7 | 21,62 | 0,002262 | | delta1+betha1 | 0,00020 |

Fig. 5. Columna F para calcular $\hat{y}_t$,.

| D | E | F | G |
|---|---|---|---|
| Y(t) | Y(t-1) | Ŷ(t) | e(t) |
| | | | |
| 0 | | | |
| 0 | 0 | 0,0001 | =D4-F4 |
| 0,58 | 0 | 0,0001 | 0,5799 |
| 21.62 | 0.58 | 0.000158 | 21.619842 |

Fig. 6. Columna G para calcular $e_t$.

**Séptimo paso:** calcular y maximizar *log L*, para estimar los parámetros del modelo. Entonces, en la Figura 8 se calcula $ln(\sigma^2_t)$ y $(e^2_t / \sigma^2_t)$, en las columnas I , J, respectivamente; luego, en la celda M11 se calcula *log L,* como se presenta en la Figura 9.

| G | H | K | L | M |
|---|---|---|---|---|
| e(t) | σ(t)^2 | | Parámetros | |
| | | | alpha0 | 0,00010 |
| | | | alpha1 | 0,00010 |
| -0,0001 | | | delta0 | 0,00010 |
| 0,5799 | =$M$4+$M$5*(G4^2)+$M$6*H4 | | | 0,00010 |
| 21,619842 | 0,00013364 | | betha1 | 0,00010 |
| 16,717738 | 0,04684177 | | delta1+betha1 | 0,00020 |
| -1,361772 | 0,02805296 | | | |

Fig. 7. Columna H del para calcular $\sigma^2_t$ del modelo GARCH

| G | H | I | J |
|---|---|---|---|
| e(t) | σ(t)^2 | log(σ(t)^2) | e(t)^2/σ(t)^2 |
| | | | |
| -0,0001 | | | |
| 0,5799 | 0,0001 | =LN(H5) | 3362,8401 |
| 21,6198 | 0,0001 | -8,9204 | 3497629,1590 |
| 16,7177 | 0,0468 | -3,0610 | =(G7^2)/H7 |
| -1,3618 | 0,0281 | -3,5737 | 66,1044 |

Fig. 8. Columna I, J para calcular $\ln(\sigma^2_t)$ y $(e^2_t / \sigma^2_t)$, respectivamente

| Optimizar | |
|---|---|
| Log L | =(-(3557/2)*LN(2*PI())-<br>0,5*SUMA(I6:I3562)-<br>0,5*SUMA(J6:J3562)) |

Fig. 9. Celda M11 para calcular *log L*.

Finalmente, se utiliza la herramienta Solver para estimar los parámetros definidos en el primer paso, maximizando el *log L* calculado en la celda M11. Los parámetros y las restricciones utilizadas son presentados en la Figura 10. En la Figura 11 se resumen los parámetros estimados. Mientas que en la Figura 12 se grafica el $\sigma^2_t$ estimado.

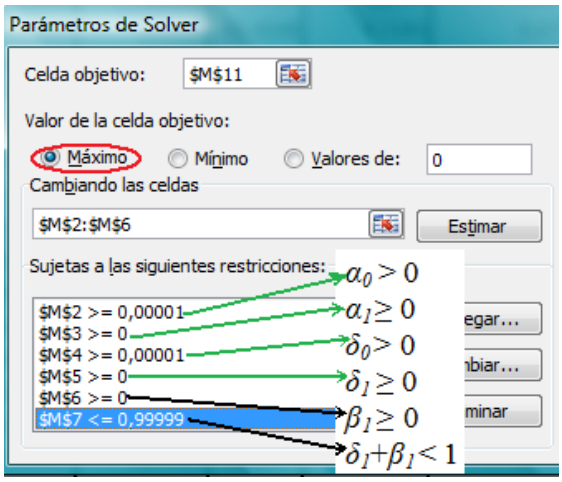


Fig. 10. Parámetros de optimización en Solver

| | L | M |
|---|---|---|
| 1 | Parámetros | |
| 2 | alpha0 | 0,00001 |
| 3 | alpha1 | 0,16820 |
| 4 | delta0 | 2,01580 |
| 5 | delta1 | 0,15864 |
| 6 | betha1 | 0,84135 |
| 7 | delta1+betha1 | 0,99999 |
| 8 | | |
| 9 | | |
| 10 | Optimizar | |
| 11 | Log L | -12844,42806 |

Fig. 11. Parámetros estimados con Solver en *Microsoft Excel*

En conclusión, la implementación realizada en esta sección sobre una hoja de Microsoft Excel presenta varias desventajas, entre ellas:

- En caso de que se desee modelar otra serie es necesario replantear toda la hoja de cálculo, dado que posiblemente cambiará la cantidad de rezagos, de datos y de parámetros, ésta dificultad podría sortearse programando el modelo con las Macros de Excel; sin embargo, el usuario estaría sometido a desarrollar en una herramienta que no es gratuita, lo que dificulta su acceso a muchas personas y empresas.

- A pesar de que Solver es una herramienta de optimización útil, en ocasiones no es robusta para realizar la optimización de múltiples parámetros, esto se refleja en que a veces es necesario asignar a los parámetros algunos puntos aleatorios, para que Solver sea capaz de encontrar una posible solución.

Sin embargo, el desarrollo del modelo en una hoja de Excel facilita la presentación del mismo, dado que permite realizar y presentar cálculos de manera sencilla y amigable.

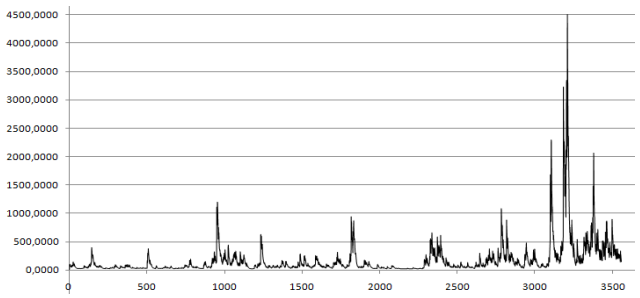


Fig. 12. Volatilidad estimada para la TRM Colombiana con un modelo GARCH(1,1) en Excel

## IV. IMPLEMENTACIÓN DE GARCH EN R

En el sistema operativo Microsoft Windows, los modelos de volatilidad pueden ser implementados en la mayoría de entornos de programación existentes, tales como: R, MATLAB, SAS, Mathematica, Visual Studio .Net, Java, etc; ó en hojas de cálculo, como: *Excel* de *Microsoft Office*, *Calc* de *OpenOffice*, *Lotus* 1-2-3 integrada en *Lotus SmartSuite*, entre otras.

Sin embargo, la licencia (derecho de uso) de la mayoría de los entornos y hojas de cálculo mencionadas no es gratuita; esto implicaría, en caso de no poseer la licencia o conjunto de licencias, un costo adicional para la empresa o persona interesada en: implementar modelos econométrico o de volatilidad; brindar o recibir servicios de consultoría y entrenamiento; y replicar las implementaciones realizadas en otras empresas o departamentos de la misma. En este caso, es recomendado utilizar software de licencia gratuita como R, *Calc* de *OpenOffice*, entre otros.

El entorno de programación de R permite crear subrutinas o funciones para resolver tareas específicas de un algoritmo determinado. Para realizar una tarea, las funciones pueden recibir datos, parámetros o argumentos; y como resultado de la tarea realizada, la función retornará una sola estructura de datos (lista, vector, matriz, etc.). Estas funciones pueden ser agrupadas en paquetes de acuerdo con una finalidad específica, por ejemplo, un paquete que agrupe varias funciones de estadística descriptiva; para más información consultar [5].

Por otro lado, sin importar el entorno de desarrollo, la implementación de los modelos es compleja por no tener pautas para diseñar e implementar su código. Dado que R es un entorno de programación gratuito y estable, se proponen las siguientes pautas para diseñar e implementar el código de un modelo GARCH en R:

- Antes de comenzar la implementación es necesario entender a cabalidad las ecuaciones del modelo.
- Crear una función por cada ecuación que tenga el modelo.
- Crear una función para optimizar y estimar los parámetros del modelo, esta puede nombrarse genéricamente *modelo.fit*, en el caso de GARCH, *Garch.fit*, en la cual se utilizarán las funciones definidas por cada ecuación del modelo.
- Crear una función para el despliegue de los resultados de la optimización del modelo, esta puede nombrarse genéricamente como *modelo.summary*, en el caso de GARCH, *Garch.summary*.
- Crear una función principal, denominada *Modelo* (es este caso *Garch*), en la cual se declaran las variables globales, se optimiza la función *modelo.fit*, y retorna el modelo estimado.
- Finalmente se crea un paquete llamado *Volatility*, que contiene las funciones creadas.

Cabe anotar que la siguiente implementación no es la única forma de realizar un modelo GARCH, y con fines didácticos las funciones se programaron utilizando sentencias y bloques de código básicos. Entonces, siguiendo las pautas dadas, las funciones necesarias para implementar un modelo GARCH son:

**Primera**, calcular $y_t$ con base en la ecuación (3); para esto, se crea la función *estimar.y* (ver Código 1), la cual recibe como parámetros la variable *alpha0* y el vector *alphas*, correspondientes a los coeficientes $\alpha_0$ y $\alpha_i$ , *i=1…p* y retorna un vector (matriz con una columna) con el valor estimado de *y* en cada instante *t* .

**Segunda**, calcular el error con base en la ecuación (2). En el Código 2, se presenta la función *calcular.error*, esta recibe el parámetro *y.estimado*; y entrega como resultado, un vector que contiene el error calculado para cada instante *t*; en el cuerpo de la función, *y.serie* es un vector global que contiene los datos reales de la serie.

```
estimar.y <- function(alpha0, alphas)
{
    y.estimado <- matrix(0,
              nrow=(length(y.serie)),
              ncol = 1)

    for (t in 2:length(y.serie))
    {
      sumatoria <- alpha0
      for (i in 1: orden.p)
      {
        if ((t - i) > 0)
        {
          sumatoria <- sumatoria +
                       (alphas[i] *
                       y.serie[t-i])
        }
      }
      y.estimado[t] <- sumatoria
    }
    return(y.estimado)
}
```

Código 1. Función para calcular $y_t$

**Tercera,** calcular la volatilidad con base en la ecuación (1). Entonces, se crea la función *estimar.sigma2* (ver Código 3) la cual recibe los parámetros necesarios para calcular $\sigma^2$ y retorna un vector con el valor $\sigma^2$ calculado para cada instante t.

```
calcular.error <- function(y.serie ,
                           y.estimado)
{
  error<-matrix(0,
              nrow=(length(y.serie)),
              ncol = 1)

  for (t in 2:length(y.serie))
  {
    error[t]<-(y.serie[t]-y.estimado[t])
  }

  return(error)
}
```

Código 2. Función para calcular $e_t$

**Cuarta,** calcular log L con base en la ecuación (4). En el Código 4 se presenta la función *calcular.logL,* esta recibe por parámetros $\sigma^2$ estimado y el error, con los cuales calcula el *log L* del modelo.

```
estimar.sigma2<-function(delta0, deltas,
                         bethas, error)
{
   sigma2.estimado<-matrix(0,
              nrow=(length(y.serie)),
              ncol = 1)

   for (t in 3:length(y.serie))
   {
     sumatoria <- delta0

     for (i in 1:orden.p)
     {
       if ((t-i)>0)
       {
         sumatoria<-sumatoria+
                (deltas[i] *
                (error[t - i] ^ 2))
       }
     }

     for (i in 1:orden.q)
     {
       if ((t-i)>0)
       {
         sumatoria<-sumatoria+
                    (bethas[i] *
              (sigma2.estimado[t - i]))
       }
     }

     sigma2.estimado[t] <- sumatoria
   }

    return(sigma2.estimado)
}
```

Código 3. Función para calcular la volatilidad

**Quinta,** realizar una función para minimizar y estimar los coeficientes del modelo, la cual deberá estar compuesta por las funciones de los Códigos 1, 2, 3, y 4. La función *Garch.fit* (Código 5) recibe un vector (*param*) con los parámetros o coeficientes que se desean optimizar, luego estos parámetros son separados en los correspondientes *alpha0, alphas, delta0, deltas y bethas*; a continuación, se calcula: *y.estimado* con la función *estimar.y;* el error con el *y.estimado*; *sigma2.estimado* con el *delta0, deltas, bethas* y el *error* calculado; finalmente, se calcula *logL* con *sigma2.estimado* y el error, y se retorna el valor de *–log L*.

```
calcular.logL<-function(sigma2.estimado,
                        error)
{
  T <- length(sigma2.estimado)

  parte1<--((T - 3) / 2) * log(2*pi)

  parte2<--0.5* sum(
              log(sigma2.estimado[3:T]))

  parte3<--0.5*sum(
       (error[3:T]^2)/
       sigma2.estimado[3:T])

  logL = parte1 + parte2 + parte3

  return(logL)
}
```

Código 4. Función para calcular log L

```
Garch.fit <- function(param)
{
  alpha0 <- param[1]
  alphas <- param[2:(orden.p + 1)]
  delta0 <- param[(orden.p + 2)]
  deltas <- param[(orden.p + 3):
                        (2*orden.p + 2)]
  bethas <- param[(2*orden.p + 3):
                  (2*orden.p + 2 + orden.q)]
  y.estimado <- estimar.y(alpha0, alphas)

  error <- calcular.error(y.estimado)

  sigma2.estimado<-estimar.sigma2(delta0,
                                  deltas,
                                  bethas,
                                  error)
  logL<-calcular.logL(sigma2.estimado, error)
  return(((-1)*logL))
}
```

Código 5. Función para ajustar los parámetros del modelo

**Sexta,** crear una función para imprimir y graficar el modelo estimado. Tal función se presenta en el Código 6, en la cual se imprimen los coeficientes y se grafica la volatilidad estimada.

**Séptima,** crear la función principal. En la función presentada en el Código 7, *Garch* recibe como parámetro la serie de tiempo $y_t$, y el orden del modelo (*p, q*); además, en ella se declaran las variables globales del modelo, se optimiza la función *Garch.fit* mediante la función *nlminb,* luego se compone el *modelo* por la lista de valores: *coef*, son los coeficientes del modelo; *sigma2,* la serie de la volatilidad estimada; *series,* la serie real; y *orden,* el orden del modelo. Finalmente, se imprime el modelo y se retorna.

```
Garch.summary <- function(model)
{
 o.p = model$orden[1]
 o.q = model$orden[2]

 alpha0<-model$coef[1]
 alphas<-model$coef[2:(o.p + 1)]
 delta0<-model$coef[(o.p + 2)]
 deltas<-model$coef[(o.p + 3):(2*o.p + 2)]
 bethas<-model$coef[(2*o.p+3):(2*o.p+2+o.q)]

 cat("Mod. Est.,Garch(p=",o.p,",q=", o.q,")")
 cat("Coeficientes \n")
 cat("Alpha0: ", alpha0, " \n")
 cat("Alphas: ", alphas, " \n")
 cat("Delta0: ", delta0, " \n")
 cat("Deltas: ", deltas, " \n")
 cat("Bethas: ", bethas, " \n\n")

 plot(model$sigma2)
}
```

Código 6. Función para presentar el modelo estimado

```
Garch <- function(y, orden = c(0,0))
{
    # Declaración Variables Globales

    # Declaración Variables
    cantVar <- 2+2*orden.p+orden.q

    # Definición de los Límites

    # Optimización
    Optim<-nlminb(rep(0.00001, cantVar),
                  Garch.fit,
                 lower = limInferior,
                 upper = limSuperior,
                  control =list( eval.max=50,
                  iter.max=50))

    modelo <- list(
                 coef   = Optim$par,
                  sigma2=ts(sigma2.estimado),
                 series = y,
                 orden  = orden
                      )

    Garch.summary(modelo)

    return(modelo)
}
```

Código 7. Función principal

La implementación realizada permite estimar modelos GARCH de cualquier orden *p, q*. Hasta este punto se tienen las funciones necesarias para el modelo, resta crear el paquete "*Volatility*", para esto es necesario cargar en R las funciones creadas y ejecutar el Código 8; este crea toda la estructura del paquete, en la ruta *C:\Volatility*, la cual contiene dos subcarpetas: *R* que contiene el código fuente de cada función; y *man* contiene la documentación de cada función, es decir, un archivo *Rd* por cada función creada; para más detalles consultar [4].

```
package.skeleton(list=c("Garch","estimar
.y","calcular.error","estimar.sigma2","c
alcular.logL","Garch.fit","Garch.summary
"), name="Volatility", path="C:/")
```

Código 8. Creación del paquete *Volatility*.

## V. PRUEBA DEL MODELO GARCH

En el Código 9 se presentan las líneas necesarias para probar el modelo *GARCH(p,q)* implementado. Primero, se leen los datos de la TRM desde un archivo de texto, estos datos se transforman en una estructura de series de tiempo mediante la función *ts*; luego, se diferencia la serie con la función *diff*; finalmente, se modela la volatilidad de la serie diferenciada con un *GARCH(1,1)*. Los parámetros estimados se presentan en la Figura 13, y la volatilidad modelada en la Figura 14, nótese que es similar a la obtenida con Excel en la Figura 12.

```
serie.dat=read.table('C:/trm.txt')
serie.ts=ts(serie.dat,names='TRM Colombia')


#Transforma la serie
serie.ts.diff = diff(serie.ts)

#calcular el Modelo
serie.ts.vol<-Garch(serie.ts.diff,c(1,1) )
```

Código 9. Creación del paquete *Volatility*.

```
Modelo Estimado, Garch( p = 1 , q = 1 )

Coeficientes
Alpha0:  1e-05
Alphas:  0.1714888
Delta0:  0.9914954
Deltas:  0.1479528
Bethas:  0.8633937
```

Fig. 13. Parámetros estimados en R, al modelar la volatilidad la TRM Colombiana con un modelo GARCH(1,1)

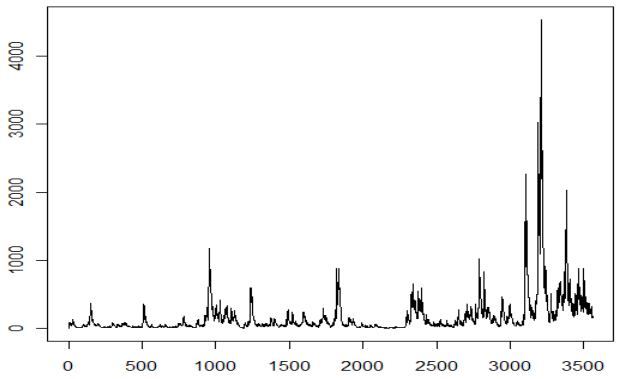


Fig. 14. Volatilidad estimada para la TRM Colombiana con un modelo GARCH(1,1) en R

## VI. CONCLUSIONES

Contar con pautas para la implementación de un modelo de volatilidad, permite estructurar el código, de tal manera que puede facilitar la programación, documentación y distribución del modelo.

Una etapa fundamental para realizar la implementación de un modelo es entender a cabalidad las ecuaciones del mismo.

La programación del modelo permite al desarrollador consolidar el conocimiento que posee sobre el mismo.

El encapsulamiento de las funciones de los modelos en paquetes facilita la distribución y mantenimiento de los mismos.

## VII. REFERENCIAS

[1] Bollerslev, T. (1986). Generalized Autoregressive Conditional Heterocedasticity. *Journal of Econometrics , 31* (3 (April)), 307-327.

[2] Casas Monsegny, M., & Cepeda, E. (2008). Modelos ARCH, GARCH y EGARCH: aplicaciones a series financieras. *Revista Cuadernos de Economía , 27* (48), 287 - 319.

[3] Engle, R. F. (1982). Autoregressive Conditional Heterocedasticity whit Estimates of the Variance of United Kingdom Inflation. *Econometrica , 50* (4), 987-1008.

[4] Development Core Team. (2009). *Writing R Extensions.* ISBN 3-900051-11-9: Version 2.10.0 (2009-10-26).

[5] Venables, W. N., & Smith, D. M. (2009). *An Introduction to R.* ISBN 3-900051-12-7 : Version 2.10.0 (2009-10-26).