UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

Paquete en R para la Estimación y Predicción en Modelos tv-Garch utilizando el algoritmo de Kalman Filter

Por: Tomás Ignacio Arancibia Beltrán

Tesis presentada a la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Concepción para optar al titulo profesional de Ingeniería Civil Matemática

8 de julio de 2025
Concepción, Chile

Profesores: Guillermo Ferreira Cabezas

Índice General

1. Introducción

En el análisis de series de tiempo se buscan modelos adecuados para la volatilidad debido a la naturaleza y la heterocedasticidad de los mercados. Uno de los modelos desarrollados fueron los modelos autorregresivos con heterocedasticidad condicional ($ARCH$), (Engle, 1982), el cual permite captar la varianza condicional de una serie de tiempo con respecto a los errores pasados. Sin embargo, este modelo presenta ciertas restricciones al momento de que la estructura de la volatilidad sea más compleja.

Para superar estas limitaciones, Bollerslev (1986) extendió lo propuesto por Engle e introdujo los modelos *Generalized ARCH* ($GARCH$), los incorporan componentes autorregresivos y de medias móviles en la varianza condicional. Estos modelos son ampliamente utilizados para estimar la volatilidad. No obstante, estos modelos aún presentan ciertas limitaciones, estas son que los parámetros son constantes en el tiempo, lo cual limita la capacidad de adaptación frente a fenómenos en los mercados financieros, también que estos modelos sobreestiman la volatilidad frente a períodos de Shock.

Teniendo en cuenta las limitaciones de los modelos $GARCH$, se desarrollaron modelos con parámetros que varían en el tiempo. Entre ellos, destacan los modelos *time-varying GARCH* (`tv-Garch`), introducidos por (Amado and Terasvirta., 2008), en modelos `tv-Garch` la varianza condicional es escrita por los autores como

$$\sigma_t^2 = g_t h_t, \quad g_t, h_t > 0,$$

donde $\{g_t\}$ es un proceso no estocástico y h_t es la especificación reescalada de un modelo $GARCH$ estacionario, por ejemplo,

$$h_t = \omega + \alpha\phi_{t-1}^2 + \beta h_{t-1}^2, \text{ donde } \phi_t^2 = u_t^2/g_t.$$

Estos modelos permiten que los coeficientes de la varianza condicional tengan una flexibilidad y adaptabilidad frente a los fenómenos presentes en los mercados financieros, debido a que los parámetros varían en el tiempo.

El método de estimación se realiza mediante Quasi máxima verosimilitud (QML). El término «quasi» significa que la estimación se realiza mediante máxima verosimilitud (ML) con base en una densidad normal, pero las estimaciones

son consistentes (sujetas a condiciones de regularidad adecuadas) incluso si la densidad de las innovaciones no es normal.

Por otro lado, (Ferreira et al., 2017) ha aplicado una herramienta estadística conocida como *Filtro de Kalman* (`KF`), el cual es un algoritmo recursivo que logra inferir los estados no observables de un sistema dinámico. Al aplicarse a modelos `tv-Garch` reformulados en espacio-estado, permite estimar la curva de los parámetros del modelo, esto mejora la capacidad del modelo en capturar la volatilidad en tiempo real. En este caso, la estimación de los parámetros, se ha utilizado la estrategia que consiste en reformular los modelos en términos de modelos *espacio-estado* a través de un filtro lineal. Esta reformulación permite describir la varianza condicional mediante un sistema de ecuaciones compuesto por una ecuación de observación y otra ecuación de estado.

El enfoque `tv-Garch` con formulación espacio-estado y aplicando el filtro de Kalman ha demostrado ser útil en diversas aplicaciones financieras. La flexibilidad para adaptarse a distintos escenarios de los mercados y su robustez en la estimación lo convierten en una herramienta poderosa en el análisis de series de tiempo financieras.

1.1. Objetivo General

Crear paquete en `R` para generar estimaciones y predicciones en modelos `tv-Garch`, enfocado en los parámetros-`tv`.

1.1.1. Objetivos específicos

- Estudio de simulación del modelo `tv-Garch` empleando Filtro de Kalman.
- Aplicación del modelo `tv-Garch` al IPSA.
- Documentación adecuada para las funciones y algoritmos implementados, junto a sus respectivos ejemplos y detalles de los parámetros y salidas.
- Creación del paquete en R para el repositorio CRAN.
- Preparar documentación y archivos necesarios para someter el paquete a proceso de revisión y finalmente posible inclusión del paquete al repositorio CRAN.

2. Marco Teórico

A lo largo de este capítulo se detallarán los conceptos básicos asociados a los distintos modelos que fueron utilizados.

2.1. Modelos

Los modelos a estudiar son los modelos `tv-Garch`, sin embargo, para esto primero se introducirán los modelos *Garch* y modelos *Arch*, los cuales dan pie para la existencia de los modelos `tv-Garch`. Estos modelos fueron surgieron debido a la necesidad de modelar de mejor forma la volatilidad en los mercados financieros.

2.1.1. Modelo *ARCH*

Los modelos autorregresivos con heterocedasticidad condicional ($ARCH$) fueron introducidos por (Engle, 1982). Las series de tiempo que siguen un proceso $ARCH$, sean ϵ_t los términos de error. Se tiene que

$$\epsilon_t = z_t \sigma_t,$$

donde, z_t es la parte estocástica y σ_t es la desviación estándar que depende del tiempo. La variable aleatoria z_t es un proceso de ruido blanco, mientras que la serie σ_t^2 se modela como:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2,$$

donde, $\alpha_0 > 0$ y $\alpha_i \geq 0,\ i > 0$. Un modelo $ARCH$(q) se puede estimar usando mínimos cuadrados ordinarios.

2.1.2. Modelo *GARCH*

Una extensión o más bien, una generalización de los modelos $ARCH$, son los modelos $GARCH$, estos fueron introducidos por (Bollerslev, 1986). Para estos modelos $GARCH$(p,q), donde p es el orden de los términos σ^2 y q es el orden para

los términos ϵ^2, viene dado por:

$$\begin{aligned} y_t &= x_t' b + \epsilon_t, \\ \epsilon|\psi_{t-1} &\sim \mathcal{N}(0, \sigma_t^2), \\ \sigma_t^2 &= \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2. \end{aligned}$$

De este tipo de modelos, existe una amplia diversidad que toman como base los modelos *GARCH*, sin embargo, no se hará enfoque a estos modelos.
Como se menciona anteriormente, estos dos modelos mencionados, *ARCH* y *GARCH*, no logran reflejar en su totalidad cambios que pueden ocurrir en el mercado financiero, para esto se trabajó con el siguiente modelo, que nos da una mejor visión al tener que los parámetros del modelo dejen de ser constantes.

2.1.3. Modelo `tv-Garch`

Los modelos `tv-Garch`, fueron introducidos por (Amado and Terasvirta., 2008), son una extensión al modelo *GARCH*, donde en estos tipos de modelos los parámetros ω, α y β son variantes en el tiempo, *time-varying*.

Sea $\{X_{t,T}\}$ una serie con media 0 y sea F_{t-1} la información disponible del pasado infinito $\{X_{t-i},\ i = 1, 2, \ldots\}$. Ahora, el modelo `tv-Garch`$(p, q)$ se define:

$$\begin{aligned} X_{t,T} &= \sigma_{t,T} \epsilon_t, \\ \sigma_{t,T}^2 &= c\left(\tfrac{t}{T}\right) + \sum_{i=1}^{p} \alpha_i \left(\tfrac{t}{T}\right) X_{t-i,T}^2 + \sum_{j=1}^{q} \beta_j \left(\tfrac{t}{T}\right) \sigma_{t-j,T}^2, \end{aligned} \tag{2.1}$$

para $t = \{1, \ldots, T\}$, donde $\sigma_{t,T}^2 = Var(X_{t,T}|F_{t-1}) = \mathbb{E}(X_{t,T}^2|F_{t-1})$ es la función de varianza condicional, $\{\epsilon_t\}$ es i.i.d. secuencia Gaussiana con media 0 y varianza unitaria y se tiene que $c(u)$, $\alpha(u)$ y $\beta(u)$ son funciones no negativas. Las funciones $c(u)$, $\alpha(u)$ y $\beta(u)$ pueden ser especificadas en términos de funciones generales. Por ejemplo, sea $\{g_j(t/T)\}$, con $j = 1, 2, \ldots$, una base de funciones suaves y sea $c(t/T)$ el parámetro variante en el tiempo del modelo (2.1). Entonces, $c(t/T)$ puede ser escrito en términos de $\{g_j(t/T)\}$:

$$c_\theta(u) = \sum_{j=0}^{k} c_j g_j(u),\ \forall u = [0, 1],$$

para valores desconocidos de k y $c_\theta = (c_0, c_1, \ldots, c_k)^\top$, de la misma forma se pueden escribir los parámetros $\alpha_i(u)$ y $\beta_j(u)$. Para trabajar con los modelos `tv-Garch`, muchos autores aproximan procesos `tv-Garch` con modelos *GARCH* estacionarios en puntos $u_0 \in (0, 1]$ cercano a t/T, (Rainer and Suhasini, 2006), (Rohan and Ramanathan, 2013), (Rohan, 2013), (Patilea and Raïssi, 2014).

En las aproximaciones existen dos tipos de escalas, una de ellas es el tiempo observado y la otra es el tiempo reescalado, la función $\sigma(u)$ está definida para el tiempo reescalado. Se puede ver que en la figura (2.1), donde ambos tiempos depende del tamaño de la muestra T. Notar que para los parámetros $c(u)$, $\alpha(u)$ y $\beta(u)$ no dependen de u, se trabajará con $u = t/T$, el cual es un reescalamiento del tiempo y trabajar con el intervalor $[0, 1]$.

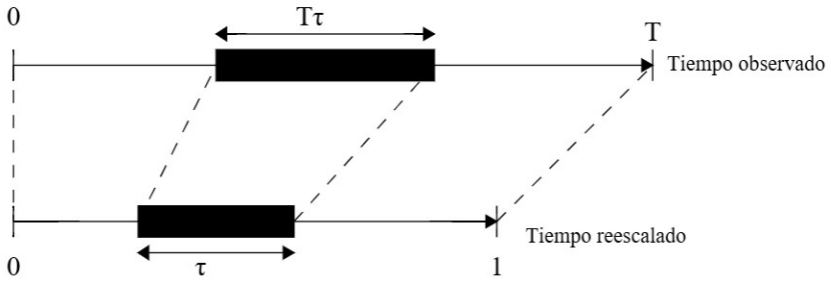

Figura 2.1: Principio de reescalado de tiempo.

Se restringe el modelo `tv-Garch`(p, q) al caso `tv-Garch`$(1, 1)$. (Rohan and Ramanathan, 2013) indica que `tv-Garch`$(1, 1)$ tiene mejores herramientas para analizar de mejor manera una data no estacionaria a diferencia de otros modelos de volatilidad. El modelo `tv-Garch`$(1, 1)$ viene dado por:

$$\begin{aligned} X_{t,T} &= \sigma_{t,T}\epsilon_t, \\ \sigma^2_{t,T} &= c(t/T) + \alpha(t/T)X^2_{t-1,T} + \beta(t/T)\sigma^2_{t-1,T}, \end{aligned} \tag{2.2}$$

donde, $c(u) = \alpha(u) = \beta(u) = 0$ para $u < 0$ y para $u \in (0, 1]$, $c(u)$, $\alpha(u)$ y $\beta(u)$ son todos positivos, estas condiciones aseguran la no negatividad de $\sigma^2_{t,T}$.

Para implementar la estimación de los parámetros *time-varying* en dominio del tiempo a través de (2.2), se asumen las siguientes condiciones de regularidad:

1. Los parámetros`tv-Garch` de (2.2) satisfacen la siguiente desigualdad:

$$0 < \alpha(u) + \beta(u) \leq 1 - \delta,\ 0 < u \leq 1 \ \textit{and}\ \sup_u\{c(u)\} < \infty,\ \textit{para algún}\ \delta > 0.$$

2. Existen finitas constantes M_1, M_2 y M_3 tal que $\forall u_1, u_2 \in (0, 1]$:

$$\begin{aligned} |c(u_1) - c(u_2)| &\leq M_1|u_1 - u_2|, \\ |\alpha(u_1) - \alpha(u_2)| &\leq M_2|u_1 - u_2|, \\ |\beta(u_1) - \beta(u_2)| &\leq M_3|u_1 - u_2|. \end{aligned}$$

La primera condición nos indica la existencia de una única solución para el proceso `tv-Garch`. La segunda condición de regularidad es una condición de contracción conocida como la continuidad de Lipschitz sobre $(0, 1]$. Estas condiciones garantizan que los parámetros *time-varying* son funciones suaves.
Notar que la dinámica de los procesos *GARCH* no es lineal y por consecuencia, la representación espacio-estado tampoco es lineal. Los algoritmos `KF` no pueden ser aplicados directamente a formulaciones de espacio-estado no lineales de modelos *GARCH*. Para trabajar este contratiempo, (Penzer, 2007), (Engle, 1982) y (Nelson and Cao, 1992), usan una aproximación para la función de la varianza condicional $\sigma^2_{t,T}$, la cual es denotada como $\tilde{\sigma}^2_{t,T}$, Ahora, se expondrán algunos ejemplos de como son los modelos *tv-GARCH*.

2.1.4. Ejemplos de Modelos `tv-Garch`

- **Ejemplo 1:** `tv-Garch(1,1)`: En este modelo, la varianza condicional está determinada por la siguiente ecuación donde los parámetros varían con el tiempo:

$$\sigma^2_t = c(t) + \alpha(t)\epsilon^2_{t-1} + \beta(t)\sigma^2_{t-1}.$$

  Luego, supongamos que los coeficientes son de la forma:

$$\begin{aligned} c(t) &= 0.05 + 0.02cos(0.01t), \\ \alpha(t) &= 0.3 + 0.1sin(0.01t), \\ \beta(t) &= 0.6 + 0.05cos(0.01t). \end{aligned}$$

Entonces, la varianza condicional queda como:

$$\sigma_t^2 = 0.05+0.02cos(0.01t)+(0.3+0.1sin(0.01t))\epsilon_{t-1}^2+(0.6+0.05cos(0.01t))\sigma_{t-1}^2.$$

- **Ejemplo 2:** `tv-Garch(2,2)`: En este modelo, la cantidad de coeficientes variables es de 5, por lo que la varianza condicional queda de la siguiente forma:

$$\sigma_t^2 = c(t) + \alpha_1(t)\epsilon_{t-1}^2 + \alpha_2(t)\epsilon_{t-2}^2 + \beta_1(t)\sigma_{t-1}^2 + \beta_2(t)\sigma_{t-2}^2.$$

Los coeficientes para este modelo toman la siguiente forma:

$$\begin{aligned} c(t) &= 0.02 + 0.001t^3, \\ \alpha_1(t) &= 0.3 - 0.005t^2, \\ \alpha_2(t) &= 0.15 + 0.04sin(0.05t), \\ \beta_1(t) &= 0.4 + 0.1cos(0.01t), \\ \beta_2(t) &= 0.2 - 0.001t. \end{aligned}$$

Así, la varianza condicional se escribe como:

$$\begin{aligned} \sigma_t^2 = {} & 0.02 + 0.001t^3 + (0.3 - 0.005t^2)\epsilon_{t-1}^2 + (0.15 + 0.04sin(0.05t))\epsilon_{t-2}^2 \\ & +(0.4 + 0.1cos(0.01t))\sigma_{t-1}^2 + (0.2 - 0.001t)\sigma_{t-2}^2. \end{aligned}$$

- **Ejemplo 3:** `tv-Garch(1,2)`: Para este ejemplo, los coeficientes variables son de un total de 4, por lo tanto, el modelo de la varianza condicional es de la siguiente forma:

$$\sigma_t^2 = c(t) + \alpha_1(t)\epsilon_{t-1}^2 + \beta_1(t)\sigma_{t-1}^2 + \beta_2(t)\sigma_{t-2}^2.$$

Utilizando $c(t)$, $\alpha_1(t)$, $\beta_1(t)$ y $\beta_2(t)$ del ejemplo 1, se tiene que la varianza condicional se escribe como:

$$\sigma_t^2 = 0.02+0.001t^3+(0.3-0.005t^2)\epsilon_{t-1}^2+(0.4+0.1cos(0.01t))\sigma_{t-1}^2+(0.2-0.001t)\sigma_{t-2}^2.$$

- **Ejemplo 4:** `tv-Garch(2,1)`: Al igual que el ejemplo anterior, tenemos un

total de 4, así, la varianza condicional toma la siguiente forma:

$$\sigma_t^2 = c(t) + \alpha_1(t)\epsilon_{t-1}^2 + \alpha_2(t)\epsilon_{t-2}^2 + \beta_1(t)\sigma_{t-1}^2.$$

Utilizando las funciones $c(t)$, $\alpha_1(t)$, $\alpha_2(t)$ y $\beta_1(t)$, se tiene que forma de la varianza condicional es:

$$\begin{aligned}\sigma_t^2 &= 0.02 + 0.001t^3 + (0.3 - 0.005t^2)\epsilon_{t-1}^2 + (0.15 + 0.04sin(0.05t))\epsilon_{t-2}^2 \\ &\quad +(0.4 + 0.1cos(0.01t))\sigma_{t-1}^2.\end{aligned}$$

2.2. Locally Stationary

En esta sección, se detallará lo que corresponde al concepto de localmente estacionario (*Locally Stationary*). Según lo propuesto por (Rainer, 1997), sea $X_{t,T}$ un proceso estocástico localmente estacionario si tiene una función de transferencia A^0 y una tendencia $\mu(\cdot)$ y puede ser representada con la siguiente forma espectral:

$$X_{t,T} = \mu\left(\frac{t}{T}\right) + \int_{\pi}^{\pi} exp(i\lambda t)A_{t,T}^0(\lambda)d\xi(\lambda), \text{ para } t = 1, \ldots, T$$

donde

- $\xi(\lambda)$ es un proceso de incremento ortogonal para el intervalo $[-\pi, \pi]$.
- Existe una constante K y una función $2\pi-$periódica $A : [0,1] \times \mathbb{R} \leftarrow \mathbb{C}$ con $A(u, -\lambda) = \overline{A(u,\lambda)}$ y

$$\sup_{t,\lambda}\left|A_{t,T}^0(\lambda) - A\left(\frac{t}{T}, \lambda\right)\right| \leq KT^{-1}, \ \forall\, T$$

  $A(u,\lambda)$ y $\mu(u)$ se asumen continuos para u.

2.3. Ecuaciones Espacio-Estado

Un modelo de espacio-estado para una serie de tiempo $\{Y_t\}$ de dimensión p consiste en dos ecuaciones, Brockwell and Davis (2002). La primera ecuación se le conoce como la ecuación de observación y la segunda ecuación se llama ecuación de estado.

$$Y_t = G_t X_t + W_t,$$

$$X_{t+1} = F_t X_t + V_t,$$

donde $\{X_t\}$ tiene dimensión m, $\{W_t\} \sim WN(0, R_t)$, $\{V_t\} \sim WN(0, Q_t)$ con dimensiones p y m, respectivamente, y $G_{t\,(p\times m)}, F_{t\,(m\times m)}$ son matrices de observación y estado, respectivamente.

2.4. Máxima Verosimilitud

El método de la máxima verosimilitud, introducido por (Fisher, 1922), es una técnica estadística usada para estimar parámetros de un modelo. Este método se basa en encontrar valores de los parámetros que maximizan la función de verosimilitud, esta mide la probabilidad de observar los datos muestrales dados ciertos parámetros. Sea una muestra de tamaño n $\{x_1, \ldots, x_n\}$ y una distribución con función de densidad $f(x;\theta)$, donde θ es el vector de parámetros desconocidos, la función de verosimilitud de define como:

$$L(\theta) = \prod_{i=1}^{n} f(x;\theta).$$

En la práctica, se maximiza la **log-verosimilitud**.

$$\ell(\theta) = \log(L(\theta)) = \sum_{i=1}^{n} \log(f(x;\theta)).$$

El estimador de la máxima verosimilitud es el valor $\widehat{\theta}$ que maximiza $\ell(\theta)$. Este estimador tiene propiedades convenientes bajo ciertas condiciones, consistencia, asíntoticamente normal y eficiente.

2.5. Filtro de Kalman

El filtro de Kalman es un algortimo matemático desarrollado por Kalman, 1960. Una de las principales características de este filtro, permite actualizar el conocimiento de una variable de estado recursiva cuando nuevos puntos en los datos están disponibles. Es decir, conociendo la distribución condicional de μ_t dado por F_{t-1} y datos nuevos y_t, sé quiere obtener la distribución condicional de μ_t dado por F_t.

Para el modelo *espacio-estado*, los predictores de un solo paso $\hat{\mathbf{X}}_t := P_{t-1}(\mathbf{X}_t)$ y

sus matrices de covarianza del error $\Omega_t = \mathbf{E}[(\mathbf{X}_t - \hat{\mathbf{X}}_t)(\mathbf{X}_t - \hat{\mathbf{X}}_t)']$ se determinan únicamente por las condiciones iniciales:

$$\hat{\mathbf{X}}_1 = P(\mathbf{X}_1|(\mathbf{Y}_0), \quad \Omega_1 = \mathbf{E}[(\mathbf{X}_1 - \hat{\mathbf{X}}_1)(\mathbf{X}_1 - \hat{\mathbf{X}}_1)'].$$

Para actualizar el estado, se utilizarán las ecuaciones recursivas definidas por:

$$\begin{aligned} \hat{\mathbf{X}}_{t+1} &= F_t\hat{\mathbf{X}}_t + \Theta_t\Delta_t^{-1}\left(\mathbf{Y}_t - G_t\hat{\mathbf{X}}_t\right), \quad t = 1, \ldots, & (2.3)\\ \Omega_{t+1} &= F_t\Omega_t F_t' + Q_t - \Theta_t\Delta_t^{-1}\Theta_t', \qquad t = 1, \ldots, & (2.4) \end{aligned}$$

donde

$$\begin{aligned} \Delta_t &= G_t\Omega_t G_t' + R_t,\\ \Theta_t &= F_t\Omega_t G_t', \end{aligned}$$

y Δ_t^{-1} es cualquier inversa generalizada de Δ_t. (Brockwell y Davis, 2016).

2.6. Representación Espacio-Estado

(Ferreira et al., 2017) propusieron un filtro para el modelo `tv-Garch`(1,1), además se indica como el filtro se puede usar para la estimación de parámetros. Para esto, se define la siguiente variable auxiliar:

$$\xi_{t,T} = c(t/T) + \beta(t/T)\xi_{t-1,T} + \alpha(t/T)\tilde{Y}_{t-1,T}, \tag{2.5}$$

donde $\xi_{t,T} = \sigma_{t,T}^2$ y $\tilde{Y}_{t,T} = X_{t,T}^2$. Por construcción, se tiene que $\xi_{t,T} = \mathbb{E}(\tilde{Y}_{t,T}|F_{t-1})$ y $\tilde{Y}_{t,T} = \xi_{t,T} + \eta_t$, donde $\{\eta_t\}$ es una serie de diferenciada de martingale. Luego, sustituyendo $\xi_{t,T} = \tilde{Y}_{t,T} - \eta_t$ en (2.5), se obtiene que:

$$\tilde{Y}_{t,T} = c(t/T) + (\beta(t/T) + \alpha(t/T))\tilde{Y}_{t-1,T} + \eta_t - \beta(t/T)\eta_{t-1}.$$

Ahora, se define la media de $\tilde{Y}_{t,T}$ por $\mu(t/T) = \mathbb{E}(\tilde{Y}_{t,T})$, lo cual satisface que $\mu(t/T) = c(t/T) + (\beta(t/T) + \alpha(t/T))\,\mu((t-1)/T)$. Luego, el modelo `tv-Garch`(1,1) en 2.2 puede ser definido por:

$$Y_{t,T} = (\beta(t/T) + \alpha(t/T))Y_{t-1,T} + \eta_t - \beta(t/T)\eta_{t-1,T}, \tag{2.6}$$

donde $Y_{t,T} = \tilde{Y}_{t,T} - \mu_{t,T}$. Se observa que (2.6) es un proceso *tv-ARMA*(1,1) que puede ser representado por un proceso espacio-estado no estacionario de la siguiente forma:

$$\begin{aligned} Y_{t,T} &= Z_{t,T}\xi_{t,T} + G_{t,T}W_{t,T}, \\ \xi_{t+1,T} &= T_{t,T}\xi_{t,T} + H_{t,T}V_{t,T}, \end{aligned} \tag{2.7}$$

donde, $\{Y_{t,T}\}$ son las observaciones,$Z_{t,T}$ es un operador de observación, $\{\xi_{t,T}\}$ es un vector de estado, $T_{t,T}$ es el operador de transición de estado y $H_{t,T}$ es un operador lineal mientras que los vectores $\{W_{t,T}\}$ y $\{V_{t,T}\}$ son independiente, donde ambos son secuencias de vectores aleatorios normales independientes los cuales tienen componentes con media cero y varianza unitaria. Además, todos los vectores involucrados en (2.7) son vectores unidimensionales. El proceso no estacionario especificado en (2.6) puede ser representado por el siguiente sistema de espacio-estado finito dimensional (ver (de Jong Piet and Jeremy, 2004)):

$$\begin{aligned} Y_{t,T} &= \xi_{t,T} + \eta_t, \\ \xi_{t+1,T} &= T_{t,T}\xi_{t,T} + H_{t,T}\eta_t, \quad t = 1, \ldots, T \end{aligned} \tag{2.8}$$

donde $Z_{t,T} = 1$, $G_{t,T} = 1$, $T_{t,T} = (\beta(t/T) + \alpha(t/T))$ y $H_{t,T} = \alpha(t/T)$.

Siguiendo la representación espacio-estado (2.8), se puede utilizar las ecuaciones del filtro de Kalman `KF` para estimar parámetros, los vectores estado, observaciones futuras y valores faltantes. Para esto, las ecuaciones `KF` son las siguientes:

$$\begin{aligned} \widehat{\xi}_{t+1,T} &= T_{t,T}\widehat{\xi}_{t,T} + \Theta_{t,T}\Upsilon_{t,T}, \\ \Omega_{t+1,T} &= T_{t,T}\Omega_{t,T}L'_{t,T} + H_{t,T}J'_{t,T}, \end{aligned} \tag{2.9}$$

donde $\Upsilon_{t,T} = Y_{t,T} - G_{t,T}\widehat{\xi}_{t,T}$ es la inovación junto a $\Delta_{t,T} = Z_{t,T}\Omega_{t,T}Z'_{t,T} + G_{t,T}G'_{t,T}$ que es la varianza, $\Theta_{t,T} = (T_{t,T}\Omega_{t,T}Z'_{t,T} + H_{t,T}G'_{t,T})\Delta^{-1}_{t,T}$, $\Omega_{t,T} = \mathbb{V}ar(\xi_{t,T} - \widehat{\xi}_{t,T})$, $L_{t,T} = T_{t,T} - \Theta_{t,T}Z_{t,T}$ y $J_{t,T} = H_{t,T} - \Theta_{t,T}G_{t,T}$. Reemplazando en (2.9), se tiene

que:

$$\begin{aligned}\widehat{\xi}_{t+1,T} &= T_{t,T}\widehat{\xi}_{t,T} + (T_{t,T}\Omega_{t,T} + H_{t,T})(\Omega_{t,T}+1)(Y_{t,T} - \widehat{\xi}_{t,T}),\\ \Omega_{t+1,T} &= T_{t,T}\Omega_{t,T}(T_{t,T} - (T_{t,T}\Omega_{t,T} + H_{t,T})(\Omega_{t,T}+1)^{-1})'\\ &\quad + H_{t,T}(H_{t,T} - (T_{t,T}\Omega_{t,T} + H_{t,T})(\Omega_{t,T}+1)^{-1}),\end{aligned}$$

Las condiciones iniciales para el filtro de Kalman son:

$$\begin{aligned}Y_{0,T} &= [0,\ldots,0],\\ \widehat{\xi}_1 &= \mathbb{E}(X_1) = [0,\ldots,0],\\ \Omega_{1,T} &= \mathbb{E}(\xi\xi^\top) = diag\{1,\ldots,1\}.\end{aligned}$$

Por lo tanto, la varianza condicional es:

$$\tilde{\sigma}_{t,T} = \widehat{\xi}_{t,T} + \frac{c(t/T)}{(1-\alpha(t/T)-\beta(t/T))}.$$

Bajo este contexto de la estimación de los parámetros, se tiene el cálculo para la estimación de la máxima verosimilitud (MLE's) por medias del sistema espacio-estado.

Dada una muestra $\{x_{t,T}\}$ de un modelo `tv-Garch`$(1,1)$ descrito por la ecuación (2.2) para $t = 1,\ldots,T$, es posible estimar el parámetro θ que máximiza la función de la verosimilitud

$$\mathcal{L}(\theta) = \sum_{t=1}^{T}[\log f_\epsilon(x_{t,T}/\tilde{\sigma}_{t,T}(\theta)) + \log(\tilde{\sigma}^2_{t,T}(\theta))],$$

donde f_ϵ es la función de densidad del error. Como se asume que $\{\epsilon_t\}$ es Gaussiano, esto permite que el vector de parámetros θ puede ser estimado maximizando:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T}[\log f_\epsilon(x^2_{t,T}/\tilde{\sigma}^2_{t,T}(\theta)) + \log(\tilde{\sigma}^2_{t,T}(\theta))] \sim \sum_{t=1}^{T}\left[\frac{\Delta^{1/2}_{t,T}x^2_{t,T}}{\tilde{\sigma}^2_{t,T}(\theta)} + \log(\tilde{\sigma}^2_{t,T}(\theta)) - \frac{\log(\Delta_{t,T})}{2}\right], \tag{2.10}$$

donde, la notación $a_T \sim b_t$ significa que $a_t/b_t \to 1$, con $T \to \infty$. Se tiene que el

MLE por KF en la ecuación (2.9) es dado por $\widehat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta)$.

2.6.1. Propiedades del Filtro Kalman en estados estacionarios.

Ahora se estudia la convergencia de KF para estados estacionarios, esto significa que las matrices $\Delta_{t,T}$, $\Theta_{t,T}$ y $\Omega_{t,T}$ se aproximan a valores constantes. Se asume que el proceso $\{Y_{t,T}\}$ satisface la ecuación (2.6) y la representación espacio-estado de la ecuación (2.8). Luego, se puede reescribir para obtener:

$$\begin{aligned} \widehat{\xi}_{t+1,T} &= (T_{t,T} - H_{t,T})\xi_{t,T} + H_{t,T}Y_{t,T} \\ &= \beta(t/T)\xi_{t,T} + H_{t,T}Y_{t,T} \\ &= \sum_{j=1}^{\infty}\prod_{k=0}^{j-1}\beta\left(\frac{t-k}{T}\right)H_{t-j,T}Y_{t-j,T} + H_{t,T}Y_{t,T}, \end{aligned} \tag{2.11}$$

recordemos que $T_{t,T} = \beta(t/T) + \alpha(t/T)$ y $H_{t,T} = \alpha(t/T)$.

Además, considerando la primera condición de regularidad, mencionada anteriormente, se tiene que $\prod_{k=0}^{n}\beta((t-k)/T) \to 0$ cuando $n \to \infty$.

Entonces, de la ecuación (2.11) se tiene:

$$\widehat{\xi}_{t,T} = \sum_{j=0}^{\infty}\psi_j\left(\frac{t}{T}\right)H_{t,T}Y_{t,T}, \tag{2.12}$$

donde los coeficientes $\psi_j\left(\frac{t}{T}\right)$ vienen dados por:

$$\psi_0\left(\frac{t}{T}\right) = 1, \text{ y } \psi_j\left(\frac{t}{T}\right) = \prod_{k=0}^{j-1}\beta\left(\frac{t-k}{T}\right)\frac{H\left(\frac{t-j}{T}\right)}{H\left(\frac{t}{T}\right)},\ j > 1.$$

Por lo tanto, tomando expectativas adicionales con respecto a la σ-álgebra $\mathcal{F}_{t-1}$ en la ecuación (2.12) se obtiene:

$$\widehat{\xi}_{t,T} - \xi_{t,T} = \sum_{j=0}^{\infty}\psi_j\left(\frac{t}{T}\right)H_{t,T}[Y_{t-j} - \mathbb{E}(Y_{t-j}|F_{t-1})],$$

con lo cual se obtiene la convergencia a 0 cuando $t \to \infty$. Entonces, $\widehat{\xi}_{t,T} \to \xi_{t,T}$ y $\Omega_{t,T} \to 0$, esto significa que la varianza del error de predicción del estado satisface

que $\Omega_{t,T} \to 0$ cuando $t \to \infty$, luego la varianza $\Delta_{t,T}$ es cercana a 1 por el largo de la muestra.

2.7. Valores faltantes y predicciones en modelos tv-Garch

Los métodos de estado-espacio y su algoritmo KF asociado tienen una metodología simple para el manejo de valores faltantes. Sea $\{Y_{t,T}\}$ un conjunto de observaciones, para $t = n+1, \ldots, T$ son los valores faltantes, entonces el vector $\Upsilon_{t,T}$ y la matriz $\Theta_{t,T}$ de KF es un conjunto de 0 para esos valores. esto es $\Upsilon_{t,T} = 0$ y $\Theta_{t,T} = 0$, entonces las ecuaciones (2.9) se tienen que:

$$\widehat{\xi}_{t+1,T} = T_{t,T}\widehat{\xi}_{t,T}, \tag{2.13}$$

$$\Omega_{t+1,T} = T_{t,T}\Omega_{t,T}T'_{t,T} + H_{t,T}H'_{t,T}, \tag{2.14}$$

donde $t = n+1, \ldots, T$.

2.7.1. Predicción a un paso

En esta sección se verá la metodología para hacer predicciones a un paso, para esto se asume que existe observaciones $\{Y_{t,T}, \ldots, Y_{n,T}\}$ del modelo espacio-estado (2.8) y queremos predecir el valor $Y_{n+1,T} = \xi_{n+1,T} + \eta_{n+1}$ y el predictor a un paso viene dado por:

$$\widehat{Y}_{n+1,T} = T_{n+1,T}\widehat{\xi}_{n+1,T},$$

donde $\widehat{\xi}_{n+1,T}$ es el estimador de $\xi_{n+1,T}$ que es obtenido por KF en la ecuación (2.13). Luego, el MSE es dado por:

$$\Delta_{n+1,T} = \Omega_{n+1,T} + 1,$$

donde $\Omega_{n+1,T}$ es dado opr la ecuación (2.14). Además, se tiene que la varianza condicional para la predicción a un paso es:

$$\tilde{\sigma}_{n+1,T} = T_{n+1,T}\widehat{\xi}_{n+1,T} + \frac{c\left(\dfrac{n+1}{T}\right)}{1 - \alpha\left(\dfrac{n+1}{T}\right) - \beta\left(\dfrac{n+1}{T}\right)}.$$

En el caso de que el predictor este fuera del tamaño de muestra $\widehat{Y}_{T+1}$ puede ser

obtenido mediante las ecuaciones KF redefiniendo el tamaño de muestra $\widetilde{T} = T + 1$ y Y_{T+1} es un valor faltante. Así, el mejor predictor cuadrático medio lineal es:

$$\widehat{Y}_{T+1,\widetilde{T}} = T_{T+1,\widetilde{T}}\widehat{\xi}_{T+1,\widetilde{T}},$$

donde $\widehat{\xi}_{T+1,\widetilde{T}}$ viene dado de la ecuación KF (2.13). Luego, el MSE viene dado por $\Delta_{T+1,\widetilde{T}} = \Omega_{T+1,\widetilde{T}}$, con $\Omega_{T+1,\widetilde{T}} = F_{T,\widetilde{T}}\Omega_{T,\widetilde{T}}F'_{T,\widetilde{T}} + Q_{T,\widetilde{T}}$.

2.7.2. Predicción a múltiples pasos

Sea $\widehat{Y}_{n+k,T} = \mathbb{E}(Y_{n+k,T}|Y_{n,T}, Y_{n-1,T}, \ldots, Y_{1,T})$ el predictor para k pasos en base al pasado $1 \leq n + k \leq T$. Siguiendo el procedimiento escrito anteriormente, las predicciones son obtenidas de las ecuaciones recursivas de Kalman dadas por las ecuaciones (2.13) y (2.14), entonces el predictor de k pasos Y_{n+k} y el MSE son dados por:

$$\widehat{Y}_{n+k,T} = \prod_{j=0}^{k} T_{n+j,T}\widehat{\xi}_{n,T}, \tag{2.15}$$

y

$$\begin{aligned}\Delta_{n+k,T} &= \prod_{j=0}^{k-1} T_{n+j,T}\Omega_{n,T}\prod_{j=0}^{k-1} T'_{n+j,T} \\ &+ \sum_{j=0}^{k-1}\prod_{i=0}^{k}(T_{n+k-i,T})^i H_{n+k-1-j,T}H'_{n+k-1-j,T}\prod_{i=0}^{k}(T'_{n+k-i,T})^i + 1,\end{aligned} \tag{2.16}$$

para $k = 1, \ldots, T - n$. Además, se tiene que la varianza condicional para las predicciones en k pasos es dado por:

$$\tilde{\sigma}_{n+k,T} = \prod_{j=0}^{k} T_{n+j,T}\widehat{\xi}_{n,T} + \frac{c\left(\dfrac{n+k}{T}\right)}{1 - \alpha\left(\dfrac{n+k}{T}\right) - \beta\left(\dfrac{n+k}{T}\right)}. \tag{2.17}$$

Bajo la suposicón (1) la predicción de la volatilidad satisface que:

$$\tilde{\sigma}_{n+k,T} \to \frac{c\left(\dfrac{n+k}{T}\right)}{1 - \alpha\left(\dfrac{n+k}{T}\right) - \beta\left(\dfrac{n+k}{T}\right)} \quad \text{con } k \to \infty.$$

El predictor de X_{n+k} viene dado por las observaciones de un proceso en el tiempo t definido por el conjunto F_{t-1}, el cual es 0 y el MSE condicional es $\mathbb{E}(\sigma^2_{t,T}|F_{t,T}) = c(t/T)/(1-\alpha(t/T)-\beta(t/T)) + \mathbb{E}(\xi_{t,T}|F_{t,T})$ para $t = 1,\ldots,n$. Luego, para obtener una aproximación del intervalo de confianza $(1-\alpha)100\,\%$, para los retornos X_{n+k} es dado por

$$[-z_{1-\alpha/2}\sqrt{\widetilde{\sigma}}_{n+k,T}, z_{1-\alpha/2}\sqrt{\widetilde{\sigma}}_{n+k,T}],$$

donde $z_{1-\alpha/2}$ es el percentil $(1-\alpha/2)$ de una distribución estándar normal.
Finalmente, para el predictor lineal en k pasos en el caso en que esté fuera del tamano de muestra, $Y_{T+k,T}$ con $k > 0$, es obtenido de la misma manera que para la predicción de un paso, esto es redefiniendo el tamao de muestra, $\widetilde{T} = T + k$ y se considera que las observaciones $\widetilde{T}+1,\ldots,\widetilde{T}+k$ son los datos faltantes.

3. Librería `tvGarchKF`

En este capítulo se presentan los códigos del paquete, además de una explicación detallada para cada uno de los códigos desarrollados para la librería `tvGarchKF`. El paquete consta con cinco scripts de R, de los cuales tres scripts son los principales para el uso de la librería, los cuales abarcan distintos aspectos para la aplicación a distintos modelos. Existe un script en lenguaje C, el cual nos reduce el tiempo de ejecución para aplicar el filtro de Kalman, con respecto a usos en scripts en R.

El primer script `"tvGarchKalmanFit"`, como se puede inducir del nombre, es el script encargado de ajustar los valores, los cuales corresponden a los coeficientes de las funciones c, α, β de los modelos `tv-Garch`.
El segundo script presente en el paquete es `"tvGarchKalmanLoglike"`, la función de este script es entregar la estimación por máxima verosimilitud, además es la función encargada para el proceso de estimación de los parámetros TV en el paso de ajuste de estos.
El tercer script que existe en el paquete es `"tvGarchKalmanPrint"`, este script cumple con el papel de entregar el modelo, es decir, nos entrega el modelo `tv-Garch` usando los datos ajustados con `"tvGarchKalmanFit"`.

Además, de la explicación a lo largo del capítulo de cada script que contiene la librería, se adjuntarán usos de la función para una comprensión más profunda a lo que respecta al uso de la librería, además se realizará una simulación para distintos casos.

En este capítulo también se verán funciones internas presentadas en otros scripts dentro del paquete que ayudan a la estimación y predicción en los modelos `tv-Garch`.

3.1. Códigos

En esta parte de la sección, se verán en detalle el funcionamiento de cada función en `R` del paquete, junto algunos ejemplos de usos de estas funciones. Además, se verá el funcionamiento de las funciones en `C` presentes en la librería. Los parámetros de entrada de cada función en `R` y `C` son expuestas en Apéndice A.

3.1.1. tvGarch_Sim

Este script de R, contiene la función la cual se puede simular una serie para su utilización posteriormente.

- **Uso:** tvGarch_Sim(series, gamma, alpha, beta, nsample, type, exponentes, trig, arg)
- **Valor de retorno:** Nos entrega una lista:
  - **x**: Esta variable representa los retornos.
  - **sigma:** Esta variable representa la varianza condicional.

```
# Ejecución
datos <- tvGarch_Sim(n = 3000, gamma = 0.05, alpha = c(0.75,
↪  0.08), beta = c(0.05, 0.03, 0.06), type =
↪  c("polynomial","polynomial","polynomial"))
# Salida
> head(datos)
                x      sigma
[1,]  0.00000000 0.00000000
[2,] -0.11855394 0.05000000
[3,]  0.23954867 0.06304390
[4,]  0.21279808 0.09619852
[5,] -0.47288494 0.08878306
[6,]  0.06980235 0.22219541
```

Código 3.1: Ejemplo de tvGarch$_{S}im$.

3.1.2. tvGarchKalmanFit

Esta función presente en la librería, como bien se dice anteriormente, es la encargada de ajustar los valores de los coeficientes de los modelos.

- **Uso:** tvGarchKalmanFit(series, c, alpha, beta, type, exponentes, trig, arg, predict)
- **Valor de retorno:**
  - Nos entrega los coeficientes ajustados.

```
# Configuración
datos <- tvGarch_Sim(n = 3000, gamma = 0.05, alpha = c(0.75,
↪  0.08), beta = c(0.05, 0.03, 0.06), type =
↪  c("polynomial","polynomial","polynomial"))
datos1 <- datos[,1]
# Ejecución
fit <- tvGarchKalmanFit(datos1, c = 0.01, alpha = c(0.5, 0.01),
↪  beta = c(0.1, 0.1, 0.05), type =
↪  c("polynomial","polynomial","polynomial"))
# Salida
> fit
# [1]  0.04968  0.75052  0.05020 -0.00381  0.11877  0.05524
```

Código 3.2: Ejemplo de tvGarchKalmanFit.

3.1.3. tvGarchKalmanLoglike

La función *tvGarchKalmanLogLike* es la encargada de trabajar en el ajuste de datos, es decir, los datos se van ajustando a medida que pasa por esta función durante la función *tvGarchKalmanFit*.

- **Uso:** tvGarchKalmanLogLike(series, c, alpha, beta, nsample, type, exponentes, trig, arg, predict)
- **Valores de retorno:** En particular, esta función no nos retorna algo en especifíco para el uso externo, más bien, esto se utiliza para el ajuste de los coeficientes.
  - Valor de la máxima verosimilitud.
  - El valor de σ^2 del error.

```
# Configuración
datos<-tvGarch_Sim(n = 3000, gamma = 0.05, alpha = c(0.75,
↪  0.08), beta = c(0.05, 0.03, 0.06),
↪  type=c("polynomial","polynomial","polynomial"))
datos1 <- datos[,1]
# Ejecución
loglike <- tvGarchKalmanLoglike(datos1, c = 0.01, alpha = c(0.5,
↪  0.01), beta = c(0.1, 0.1, 0.05), type =
↪  c("polynomial","polynomial","polynomial"))
# Salida
> loglike
# [1] -2794.988
# attr(,"sigma")
# [1] 0.19618243 0.04939829 0.05194918 0.05827559 0.13751080
↪  0.35385011 0.30192026 0.17431459 0.16421166 0.38041225
↪  0.21792478 0.05058389
# [13] 0.19566923 0.05152897 0.05961790 0.05902080 0.05980434
↪  0.05523200 0.06676610 0.07790580
# [ reached getOption("max.print") -- omitted 2981 entries ]
```

Código 3.3: Ejemplo de tvGarchKalmanLoglike.

3.1.4. tvGarchKalmanPrint

Esta función es la encargada de entregarnos el modelo solicitado.

- **Uso:** tvGarchKalmanPrint(series, c, alpha, beta, nsample, type, exponentes, trig, arg, predict)
- **Valor de retorno:** Nos entrega una lista:
  - **X**: Ecuación estado de filtro de Kalman.
  - **Sigma:** Son los valores de la varianza condicional.
  - **Fm**: Valor del MSE.
  - **Loglike:** Es el valor de la máxima verosimilitud.

```
# Configuración
datos <- tvGarch_Sim(n = 3000, gamma = 0.05, alpha = c(0.75,
↪  0.08), beta = c(0.05, 0.03, 0.06), type =
↪  c("polynomial","polynomial","polynomial"))
datos1 <- datos[,1]
fit <- tvGarchKalmanFit(datos1, c = 0.01, alpha = c(0.5, 0.01),
↪  beta = c(0.1, 0.1, 0.05), type =
↪  c("polynomial","polynomial","polynomial"))
# Ejecución
model <- tvGarchKalmanPrint(fit, datos1, c = 0.01, alpha =
↪  c(0.5, 0.01), beta = c(0.1, 0.1, 0.05), type =
↪  c("polynomial","polynomial","polynomial"))
# Salida
> lapply(model, head, 10) # Se usa lapply junto a la función
↪  head para ver solamente los 10 primeros valores de la lista
↪  de salida.
$sigma
# [1] 0.4277172 0.2355376 0.2788616 0.2338956 0.2899586
↪  0.2372237 0.2454230 0.3694270 0.4626722 0.3083840
$X
# [1]  0.00000000 -0.12750997 -0.10527010 -0.12837272
↪  -0.09904988 -0.12689678 -0.12298548 -0.04678768  0.03075552
↪  -0.08825551
$Fm
# [1] 1.948746 1.002037 1.000009 1.000000 1.000000 1.000000
↪  1.000000 1.000000 1.000000 1.000000
$loglik
# [1] -3027.037
```

Código 3.4: Ejemplo de tvGarchKalmanPrint.

3.1.5. tvGarchKF.c

Esta sección se verá el contenido del script en C presente en el paquete. Este script contiene 2 funciones: ***tvGarch1ab*** y ***tvGarch1ab2***, las cuales tienen un funcionamiento similar. Estas funciones son las encargadas de actualizar las ecuaciones recursivas de Kalman (2.9), donde ***tvGarch1ab*** es para cuando el parámetro del modelo `tv-Garch` $c(u)$ es una función constante, en otro caso, se utiliza la función ***tvGarch1ab2***. A continuación, se verán detalles de ambas funciones:

- ***tvGarch1ab***: Esta función se utiliza en el caso que la función $c(u)$ del modelo `tv-Garch` sea constante.
  - **Uso**: El uso de esta función para su ejecución en R es la siguiente: .C("tvGarch1ab", as.integer(nsample), as.double(a), as.double(b), as.double(c), as.double(P), as.double(X), as.double(F.m), as.double(K), as.double(y)).
- ***tvGarch1ab2***: Esta función se utiliza en el caso que la función $c(u)$ del modelo `tv-Garch` no sea constante.
  - **Uso**: El uso de esta función para su ejecución en R es la siguiente: .C("tvGarch1ab2", as.integer(nsample), as.double(a), as.double(b), as.double(c), as.double(P), as.double(X), as.double(F.m), as.double(K), as.double(y)).

Para ambas funciones los valores de retornos son las mismas de entrada, es decir, la función nos entrega las mismas variables que le entregamos, la diferencia, es que se actualizarán las variables X, $F.m$ y K, que son las variables de interés, las otras variables al no actualizarse la salida de estas son las mismas que se le entrega a la función.

```
library("tvGarchKF")
# Configuración
series <- tvGarch_Sim( n = 3000 , gamma = 0.05 , alpha = c(0.75
↪  , 0.08) , beta = c(0.05 , 0.03 , 0.06) , type = c(
↪  "polynomial","polynomial","polynomial"))[ ,1]
nsample = length(series)
u<-(1:nsample)/nsample
c = 0.05;a = 0.75 + 0.08*u;b = 0.05 + 0.03*u + 0.06*u^2
y<-NULL
for(i in 1:nsample){y[i]<-ifelse (is.na(series[i]), 100000000,
↪  series[i]^2-c/(1-a[i]-b[i]))}
P <-rep(0, nsample);X <-rep(0, nsample)
F.m <-rep(0, nsample);K<-rep(0, nsample)
# Inicializo P_1
P[1]<-a[1]^2/(1-(a[1] + b[1])^2)

# Ejecución
resultado<-.C("tvGarch1ab", as.integer(nsample), as.double(a),
↪  as.double(b), as.double(c), as.double(P), as.double(X),
↪  as.double(F.m), as.double(K), as.double(y))
X=resultado[[6]]
F.m=resultado[[7]]
K=resultado[[8]]

# Salida
> head(X)
[1]  0.00000000 -0.19516663 -0.19023394 -0.19712125 -0.09137425
↪  -0.03609198
> head(F.m)
[1] 2.562866 1.001525 1.000004 1.000000 1.000000 1.000000
> head(K)
[1] 0.7805234 0.7501295 0.7500802 0.7501067 0.7501333 0.7501600
```

Código 3.5: Ejemplo de `tvGarch1ab` ejecutado desde R.

```
library("tvGarchKF")
# Configuración
series <- tvGarch_Sim( n = 3000 , gamma = c(0.05,0.03) , alpha
↪  = c(0.75, 0.08,0.02) , beta = c(0.05 , 0.03 , 0.06) , type
↪  = c( "polynomial","polynomial","polynomial"))[ ,1]
nsample = length(series)
u<-(1:nsample)/nsample
c = 0.05 + 0.03*u;a = 0.75 + 0.08*u + 0.02*u^2;b = 0.05 +
↪  0.03*u + 0.06*u^2
y<-NULL
for(i in 1:nsample){y[i]<-ifelse (is.na(series[i]), 100000000,
↪  series[i]^2-c[i]/(1-a[i]-b[i]))}
P <-rep(0, nsample);X <-rep(0, nsample)
F.m <-rep(0, nsample);K<-rep(0, nsample)

# Inicializo P_1
P[1]<-a[1]^2/(1-(a[1] + b[1])^2)

# Ejecución
resultado<-.C("tvGarch1ab2", as.integer(nsample), as.double(a),
↪  as.double(b), as.double(c), as.double(P), as.double(X),
↪  as.double(F.m), as.double(K), as.double(y))
X=resultado[[6]]
F.m=resultado[[7]]
K=resultado[[8]]

# Salida
> head(X)
[1]  0.00000000 -0.19516663 -0.19023394 -0.19712125 -0.09137425
↪  -0.03609198
> head(F.m)
[1] 2.562866 1.001525 1.000004 1.000000 1.000000 1.000000
> head(K)
[1] 0.7805234 0.7501295 0.7500802 0.7501067 0.7501333 0.7501600
```

Código 3.6: Ejemplo de `tvGarch1ab2` ejecutado desde R.

3.2. Simulación

En esta sección se verán simulaciones para distintos modelos proporcionados en (Ferreira et al., 2017). Antes de detallar sobre las simulaciones y los modelos, mencionar que para las simulaciones se usa el software R y algunas funciones en C las cuales se conectan a R. Se usa la función *optim*, debido a que optimiza de mejor manera la función y además trabaja con la función "BFGS" que corresponde a al método quasi-Newton (Nocedal and Wright, 2006) y "CG" corresponde al método del gradiente conjugado (Fletcher and Reeves, 1964). El primero de estos se utiliza al momento de haber modelos que contienen una estructura no lineal y en el caso lineal se opta por la segunda opción.

Las funciones en C, no están resueltas en R, puesto que, el proceso de utilizar estas mismas funciones en R demora un tiempo más largo que en C. Se tiene que para una muestra finita utilizando el procedimiento `KF` se analiza a través de experimentos de Monte Carlo, lo cual exige mucho tiempo de cálculo. Por ejemplo, para una muestra de largo $T = 3000$ la diferencia es de aproximadamente 12 segundos, lo cual es considerable a la hora de trabajar o de realizar simulaciones, esto se encuentra en (Ferreira et al., 2017).
Como se menciona en la sección (2.1.3) el capítulo (2), se realizarán simulaciones con el modelo `tv-Garch`$(1, 1)$ que se analizarán calculando la media en experimentos de Monte Carlo. El modelo en cuestión puede ser descrito como en la ecuación (2.2). La estimación de los parámetros variantes en el tiempo, *time-varying*, $\theta(u) = (c(u), \alpha(u), \beta(u))$, son obtenidos a través del algoritmo `KF` usando el procedimiento mencionado en la sección (2).

Se trabajará con distintas estructuras para los parámetros *time-varying* de la ecuación (2.2), para esto se consideran los siguientes 3 modelos, con distintas combinaciones para $\theta(u)$, entre estos estan lineales, no lineales y cuadráticos.

Modelo 1: $c(u) = c_0$, $\alpha(u) = a_0 + a_1 u$, $\beta(u) = b_0 + b_1 u$,
Modelo 2: $c(u) = c_0$, $\alpha(u) = a_0 + a_1 u$, $\beta(u) = b_0 + b_1 u + b_2 u^2$,
Modelo 3: $c(u) = c_0 + c_1 u$, $\alpha(u) = a_0 + a_1 \sin(\pi/4u)$, $\beta(u) = b_0 + b_1 u + b_2\sqrt{u}$,

para $u \in [0, 1]$. El número total de repeticiones de Monte Carlo son 1000 para cada tamaño de muestra. Para el modelo 3, se considero para los parámetros

time-varying valores sugeridos por los autores (Carnero et al., 2012) y (Lorenzo et al., 2006), $c \in [0.02, 0.1]$, $\alpha \in [0.05, 0.15]$ y $\beta \in [0.8, 0.88]$.

3.2.1. Resultado de las simulaciones

Los resultados para las simulaciones de los 3 modelos propuestos para distintos tamaños de muestras, estos resultados están en la tabla 3.1. Podemos notar que para muestras de tamaño $T = 3000$ se obtiene una aceptable estimación para los modelos 2 y 3. Para el modelo 1 las aproximaciones son buenas a bajos valores de T, a pesar de haber unas diferencias grandes entre algunos valores reales y los valores estimados.

Tabla 3.1: Resultados de las simulaciones realizadas para los 3 modelos.

			Tamaños de muestra					
Modelo			T = 500	T = 1000	T = 1500	T = 3000	T = 4000	T = 6000
Modelo 1	Coeficiente Real							
c_0	0.05	$\hat{c_0}$	0.0747	0.0692	0.0658	0.0661	0.0635	0.0624
a_0	0.05	$\hat{a_0}$	0.0871	0.0825	0.0829	0.0875	0.0814	0.0830
a_1	0.15	$\hat{a_1}$	0.0932	0.1017	0.1032	0.1041	0.1076	0.1070
b_0	0.75	$\hat{b_0}$	0.6171	0.6455	0.6589	0.6546	0.6699	0.6732
b_1	0.05	$\hat{b_1}$	0.1673	0.1446	0.1346	0.1390	0.1276	0.1252
		$\hat{\sigma}(\hat{c_0})$	0.0302	0.0217	0.0174	0.0131	0.0116	0.0113
		$\hat{\sigma}(\hat{a_0})$	0.0687	0.0483	0.0392	0.0299	0.0255	0.0241
		$\hat{\sigma}(\hat{a_1})$	0.1211	0.0797	0.0649	0.0445	0.0387	0.0311
		$\hat{\sigma}(\hat{b_0})$	0.1056	0.0847	0.0718	0.0579	0.0518	0.0542
		$\hat{\sigma}(\hat{b_1})$	0.1033	0.0782	0.0647	0.0482	0.0443	0.0415
Modelo 2	Coeficiente Real							
c_0	0.05	$\hat{c_0}$	0.0500	0.0499	0.0500	0.0499	0.0500	0.0500
a_0	0.75	$\hat{a_0}$	0.7062	0.7242	0.7261	0.7321	0.7340	0.7356
a_1	0.08	$\hat{a_1}$	0.1000	0.0909	0.0970	0.0978	0.0963	0.0978
b_0	0.05	$\hat{b_0}$	0.0454	0.0458	0.0460	0.0489	0.0468	0.0471
b_1	0.03	$\hat{b_1}$	-0.0167	-0.0038	0.0085	0.0135	0.0166	0.0210
b_2	0.06	$\hat{b_2}$	0.0710	0.0708	0.0634	0.0613	0.0627	0.0587
		$\hat{\sigma}(\hat{c_0})$	0.0094	0.0064	0.0048	0.0033	0.0029	0.0023
		$\hat{\sigma}(\hat{a_0})$	0.1091	0.0823	0.0596	0.0481	0.0397	0.0359
		$\hat{\sigma}(\hat{a_1})$	0.1281	0.1042	0.0745	0.0573	0.0492	0.0511
		$\hat{\sigma}(\hat{b_0})$	0.0750	0.0507	0.0400	0.0276	0.0236	0.0207
		$\hat{\sigma}(\hat{b_1})$	0.0991	0.0775	0.0592	0.0424	0.0384	0.0351
		$\hat{\sigma}(\hat{b_2})$	0.1003	0.0797	0.0623	0.0429	0.0371	0.0367
Modelo 3	Coeficiente Real							
c_0	0.02	$\hat{c_0}$	0.0223	0.0209	0.0209	0.0205	0.0201	0.0206
c_1	0.08	$\hat{c_1}$	0.0741	0.0754	0.0774	0.0782	0.0788	0.0791
a_0	0.88	$\hat{a_0}$	0.8092	0.8247	0.8362	0.8582	0.8694	0.8682
a_1	-0.10	$\hat{a_1}$	-0.0525	-0.0136	-0.0290	-0.0735	-0.0875	-0.0786
b_0	0.05	$\hat{b_0}$	-0.1130	-0.0627	-0.0274	0.0111	0.0187	0.0255
b_1	0.35	$\hat{b_1}$	-0.0743	0.0135	0.0933	0.2131	0.2325	0.2548
b_2	-0.25	$\hat{b_2}$	0.2817	0.1621	0.0485	-0.0907	-0.1155	-0.1436
		$\hat{\sigma}(\hat{c_0})$	0.0128	0.0087	0.0061	0.0046	0.0035	0.0027
		$\hat{\sigma}(\hat{c_1})$	0.0364	0.0237	0.0184	0.0151	0.0107	0.0087
		$\hat{\sigma}(\hat{a_0})$	0.1758	0.1274	0.1048	0.0789	0.0706	0.0595
		$\hat{\sigma}(\hat{a_1})$	0.3918	0.2590	0.2182	0.1664	0.1455	0.1239
		$\hat{\sigma}(\hat{b_0})$	0.2449	0.1532	0.1149	0.0798	0.0621	0.0532
		$\hat{\sigma}(\hat{b_1})$	0.5087	0.3576	0.2852	0.2235	0.1760	0.1492
		$\hat{\sigma}(\hat{b_2})$	0.6903	0.4652	0.3592	0.2720	0.2053	0.1706

3.3. Aplicación

En esta sección se verá una aplicación de la metodología propuesta para el modelamiento de la estimación de los parámetros y predicciones a datos reales con retornos financieros. Se usarán técnicas desarrolladas en la sección 2.6 para el análisis de una data financiera que se encuentra comúnmente en la literatura financiera con un gran impacto en su mercado.

3.3.1. Índice de Precios Selectivo de Acciones mensual (IPSA)

Se analizará el retorno del Índice de Precios Selectivo de Acciones mensuales (IPSA). Se considera los retornos diarios del IPSA, desde el 1 de Marzo del 2000 hasta el 14 de Octubre del 2014. Esta data esta disponible en https://es.investing.com/indices/ipsa o https://finance.yahoo.com/quote/%5EIPSA/. Para encontrar la

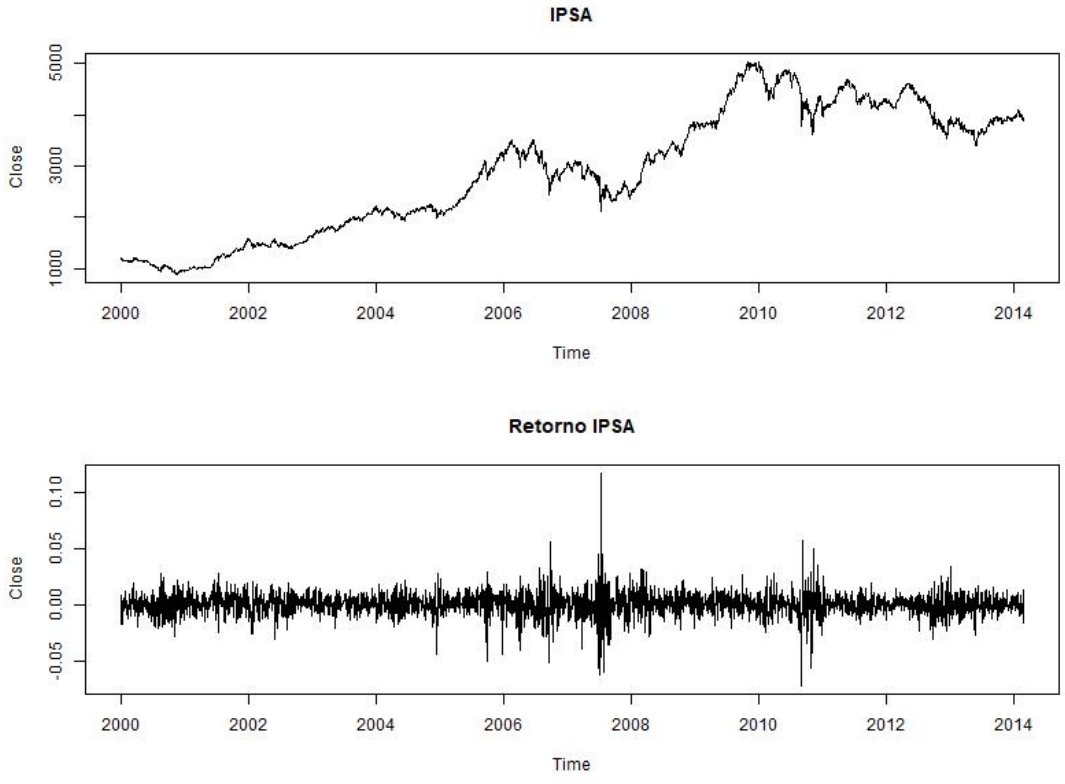

Figura 3.1: IPSA. El gráfico superior corresponde a los datos del IPSA. El segundo gráfico corresponde a los retornos IPSA.

estructura dinámica de los parámetros, se dispone a usar el análisis para datos descrito en la sección 2 usando $N = 800$ observaciones en cada una de las $M = 24$

ventanas con un salto de $S = 100$ observaciones. Dentro de cada una de estas ventanas, los parámetros de un proceso $GARCH(1,1)$ estacionario fueron estimados usando la librería `fGarch`.

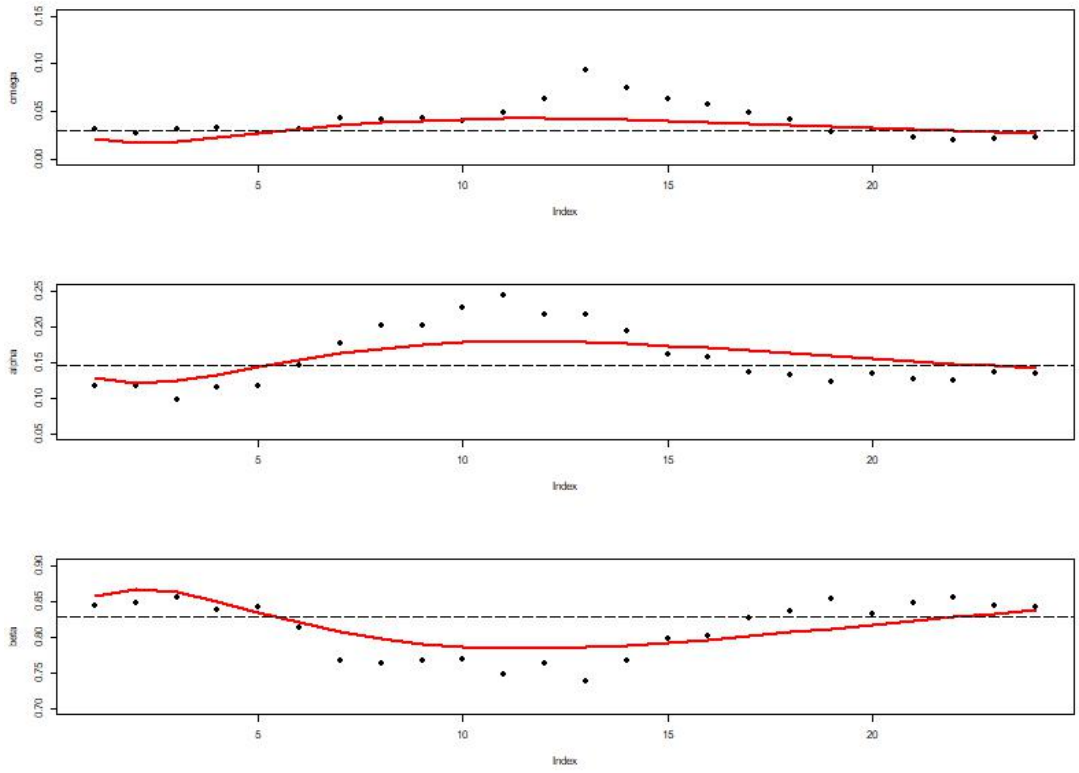

Figura 3.2: Retornos IPSA. (a) Estimación de $c(u)$. (b) Estimación de $\alpha(u)$. (c) Estimación de $\beta(u)$. En los 3 gráficos, se tiene la línea punteada es el modelo $GARCH$ estacionario, la línea roja continua representa el modelo `tv-Garch` y los puntos representa una aproximación heurística.

La figura 3.2 muestra la estimación heurística representada por los puntos. Además, la línea horizontal punteada nos indica el modelo *GARCH* estacionario con los parámetros estimados de (c, α, β) considerando toda la información de los datos, mientras que la línea continua roja indica las funciones *time-varying* de (c, α, β) representando el modelo `tv-Garch` ajustado.
Notamos que en la figura 3.2 nos señala considerar las siguientes funciones para $c(u)$, $\alpha(u)$ y $\beta(u)$:

$$c(u) = c_0 + c_1 \cos(3(1 - \log(u))),\ \alpha(u) = \alpha_0 + \alpha_1 \cos(3(1 - \log(u))),$$
$$\beta(u) = \beta_0 + \beta_1 \cos(3(1 - \log(u))), \quad \text{para } u \in [0, 1]. \tag{3.1}$$

En la tabla 3.2 se evidencia la estimación de los parámetros usando el procedimiento `KF` para el modelo `tv-Garch`$(1, 1)$ con una estructura *time-varying* dado por la ecuación (3.1), además en la tabla 3.2 se encuentran valores para la desviación estándar y se aplicó un t-test, los cuales fueron obtenidos a través de series simulaciones de largo igual a los datos usados del IPSA, se hicieron 1000 repeteciones Monte Carlo para la obtención de los resultados.

Tabla 3.2: Retornos del IPSA: parámetros estimados con un modelo `tv-Garch`$(1, 1)$ dado por (3.1)

Parameter	Estimates	Desviación Estándar	t-value
c_0	0.03001	0.1256	39.7426
c_1	0.01237	0.1224	-8.4833
α_0	0.14990	0.1246	67.2791
α_1	0.02919	0.1705	-14.0967
β_0	0.82541	0.2873	59.5648
β_1	-0.04145	0.1143	6.5539

En la figura 3.3 muestra la estimación de la volatilidad en el modelo `tv-Garch`$(1, 1)$ ajustado. Es importante notar que la estimación de la volatilidad captura de buena manera la volatilidad de los retornos de los datos del IPSA.

Finalmente, se construye un intervalo de predicción para los retornos y la volatilidad de los precios de cierre diarios del IPSA. Se considera una ventana de los datos que consiste en remover 20 observaciones de la muestra. Esta ventana que se remueve considera los datos $t = 3165, \ldots, 3185$.
La figura 3.4 muestra las predicciones para la volatilidad $\hat{\sigma}^2_{3165+k,T}$ y para los

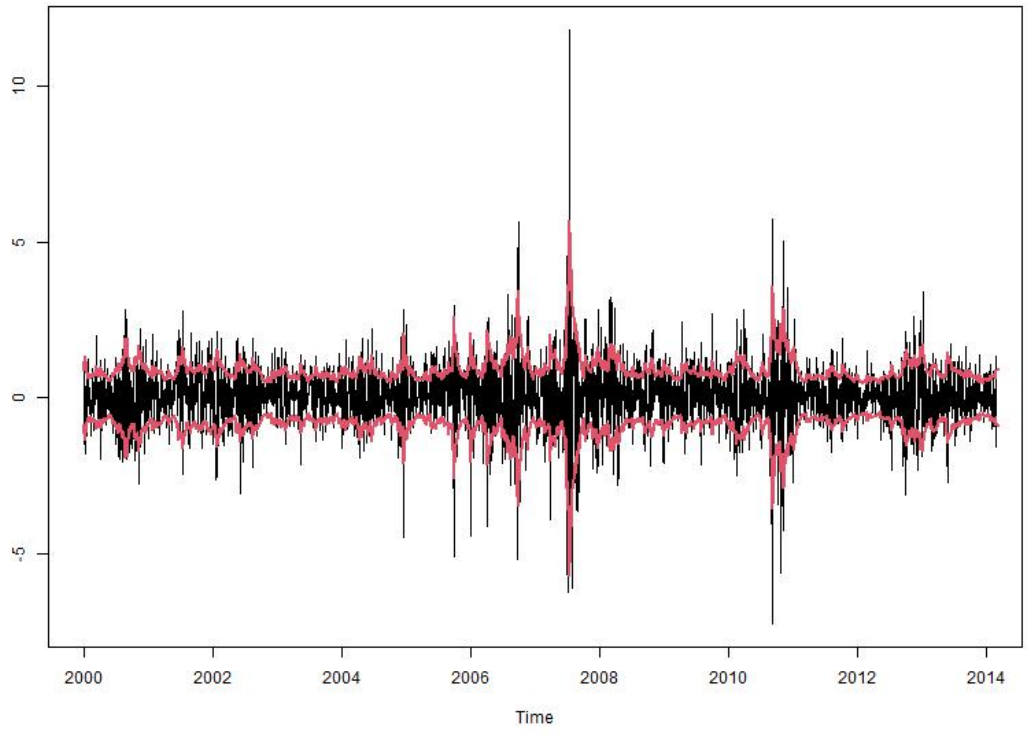

Figura 3.3: Retornos IPSA: El color negro representa a los retornos de la serie y las líneas rojas continuas representan la estimación de la volatilidad.

retornos $\hat{X}^2_{3165+k,T}$, donde $k = 1, \ldots, 20$ y su respectiva bandas de predicción al 95 %. Se observa que las predicciones se encuentran dentro de las bandas y destacar que las bandas de línea continua corresponden a la predicción bajo el contexto *time-varying* y las bandas de línea punteada corresponden al caso estacionario.

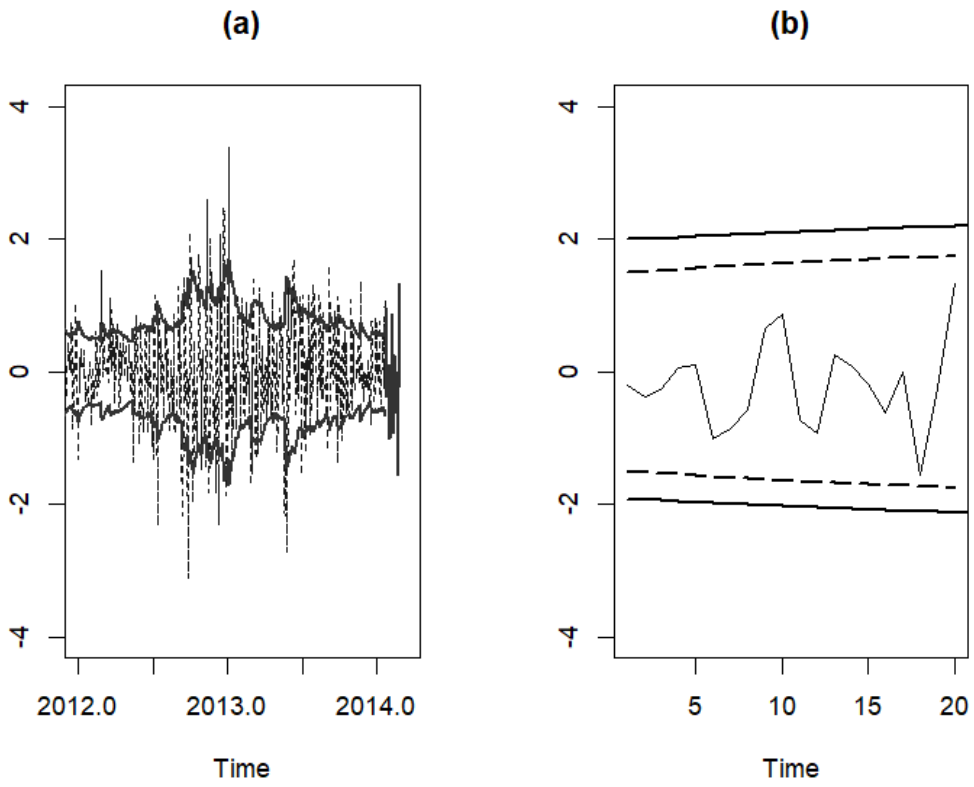

Figura 3.4: Retornos IPSA. Predicciones a múltiples pasos de valores faltantes y bandas de predicción de 95 %, para la volatilidad (a) y para los retornos (b).

Se realizaran dos comparaciones sobre el modelo propuesto, una comparación es a un modelo $GARCH(1,1)$ estacionario y la otra comparación es con un modelo `tv-Garch`(1,1) estacionario de la librería *tvgarch* que esta en el repositorio CRAN. Se presentaran distinta tablas haciendo las dos comparaciones mencionadas, una sección corresponde a una comparación dentro de la muestra y otra sección a fuera de la muestra, es decir, predicciones.

3.3.2. Dentro de la muestra

Tabla 3.3: Comparación de modelos dentro de la muestra.

	ME	RMSE	MAE
Modelo propuesto	-0.9097	1.4330	1.0620
GARCH Estacionario	-0.9055	1.4292	1.0591
`tvGarch`(1,1) Estacionario	-0.9063	1.4308	1.0598

3.3.3. Fuera de la muestra

Tabla 3.4: Comparación de modelos fuera de la muestra.

	ME	RMSE	MAE
Modelo propuesto	-0.2357	0.4834	0.4164
GARCH Estacionario	-0.2858	0.5092	0.4359
`tv-Garch`(1,1) Estacionario	-0.7428	0.9953	0.8597

En la tabla (3.3), se aprecia que los modelos son similares, con una ligera ventaja el modelo GARCH estacionario. Sin embargo, al observar la tabla (3.4), se tiene que el modelo propuesto presente una ventaja por sobre los otros dos modelos, por lo que se puede interpretar que el modelo propuesto genera mejores predicciones que otros modelos.

4. Documentación

El desarrollo de la librería *tvGarchKF* fue dirigido por el manual (R Core Team, 2024), esto para poder seguir las condiciones para que la librería sea aceptada por el repositorio *CRAN*. En este capítulo de desarrollará la creación del paquete, detallando el paso a paso, describiendo los importantes archivos presentes y las carpetas involucradas dentro de la librería, donde se detallarán, `DESCRIPTION` el cual contiene la metadata de la librería, es decir, especificaciones del paquete, la carpeta *R*, que contiene las funciones, los scripts, *data* contiene el conjunto de datos para los ejemplos dentro de la librería, *man* que alberga la documentación de las funciones, el archivo `NAMESPACE`, que define las funciones utilizadas en el paquete, además de contener también la exportación de librerías y/o funciones utilizadas de otras librerías, luego, la carpeta *src*, la cual contiene las funciones en lenguaje `c` utilizadas para en procesos dentro la librería y por último, se tenía la carpeta *inst* la cual nos entrega la manera correcta de citar el paquete. Los contenidos de cada archivo pueden ser obtenidos en el `Github:` https://github.com/Tomas-UDEC/tvGarchKF.

El proceso comienza con la creación del paquete, el cual es el paso inicial para comenzar, esto se detallará en la sección *Configuración Inicial* (4.1), donde se describe el paso a paso para preparar los archivos y el entorno. Seguido de esta etapa inicial, se verán el resto de los archivos y carpetas mencionados anteriormente.

4.1. Configuración Inicial

Para crear el paquete en **R**, el primer paso es abrir el programa `RStudio` y crear un nuevo proyecto en la esquina superior izquierda en la pestaña *File* y luego, en el apartado de *New Project* para crear el nuevo proyecto para nuestro paquete, esto nos conducirá a la ventana mostrada en la figura 4.1, aquí se selecciona la ubicación en donde se encontrará el nuevo proyecto, o bien en este caso, nuevo paquete.

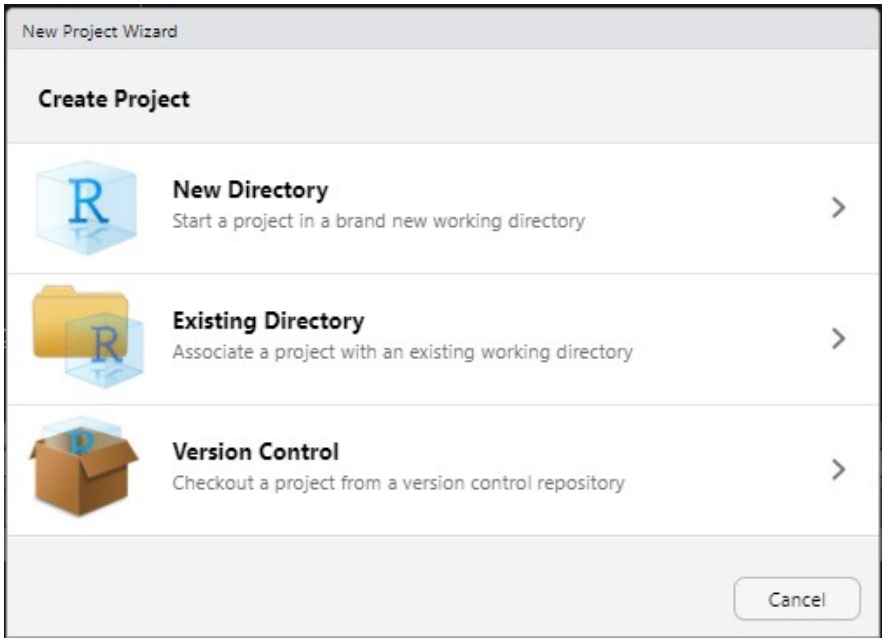

Figura 4.1: Nuevo proyecto en `RStudio`, se selecciona el directorio para el nuevo proyecto.

Para evitar confusiones con la ubicación del paquete, lo recomendado es elegir la opción **‘`New Directory`’**. Luego, tendremos la ventana de la figura 4.2, en donde tenemos que seleccionar el tipo de proyecto que queremos realizar.

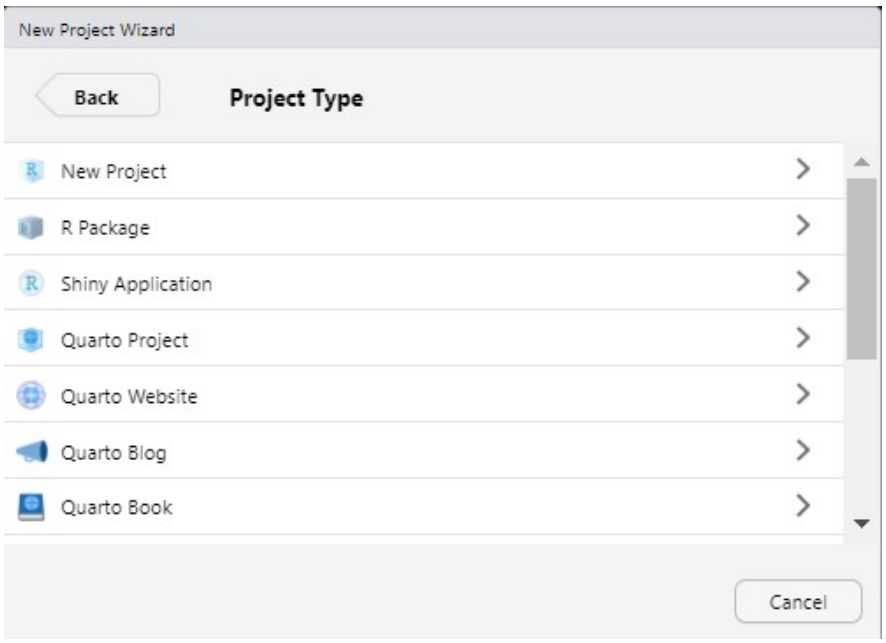

Figura 4.2: Distintos tipos para el nuevo proyecto.

Al seleccionar ‘‘`R Package`’’ en la figura 4.2, nos llevará a la ventana figura 4.3 donde se observa la configuración inicial para el nuevo paquete, el nombre, el tipo de paquete y la ubicación de donde se guardará localmente el paquete.

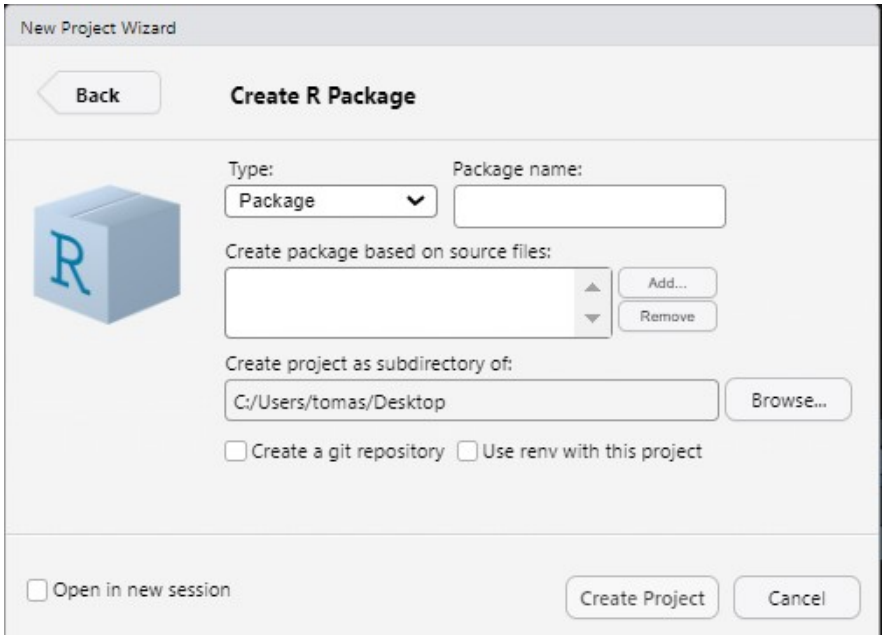

Figura 4.3: Configuración inicial para el proyecto, en este caso, nuevo paquete en `RStudio`.

Una vez completados los campos requeridos, se selecciona ‘‘`Create Project`’’ y estaría listo la creación inicial, al momento de seleccionar está opción en la ubicación del paquete se crearán distintos archivos y carpetas que se verán a los largo de este capítulo, entre estos archivos están el `DESCRIPTION` y `NAMESPACE` que son fundamentales para el paquete, además se crea un archivo `.Rproj` que esta encargado de la gestión del entorno y facilita la carga del proyecto. Estos pasos aseguran que la configuración inicial este creada de manera correcta para el desarrollo posterior.

4.2. Description

Siguiendo el manual (R Core Team, 2024) se tiene que el archivo `DESCRIPTION` es esencial para cualquier paquete en **R**, este archivo actúa como presentación para el paquete, contiene los metadatos del paquete, tales como, autores, una breve descripción del paquete, fecha, entre otros. Ahora, se detallarán los campos empleados en este archivo para el paquete `tvGarchKF`:

- **Package:** Este campo hace referencia al nombre único del paquete para el CRAN, en este caso es `tvGarchKF`.
- **Date:** Es la fecha publicación de la versión actual.
- **Type:** Se establece como "package"para indicar que es un paquete de **R**.
- **Title:** En este campo se da una breve descripción del paquete.
- **Version:** Es la versión del paquete.
- **AuthorsR** Son las personas involucradas en la creación del paquete junto a sus correos electrónicos, en este caso se incluyen a Guillermo Ferreira y Tomás Arancibia, como los autores del paquete.
- **Maintener:** Designa quien esta a cargo para mantener actualizado el paquete, en este caso Tomás Arancibia, es quien asume esta responsabilidad, facilitando su correo electrónico.
- **Description:** Contiene la descripción del paquete de manera detallada y en un solo párrafo, para este paquete, se tiene una explicación detallada del funcionamiento y propósito del paquete.
- **Reference:** Incluye referencias de trabajos de investigación y/o publicaciones relevantes para el paquete, (Ferreira et al., 2017), es donde se respalda el contenido del paquete.
- **License:** Se especifica bajo que licencia opera el paquete, GPL(>= 3).
- **Encoding:** Es la codificación de carácteres que usa el paquete, en este caso UTF-8.
- **LazyData:** Este campo es rellenado con TRUE, para obtener que los datos no sean cargados innecesariamente y solo cuando sean requeridos.
- **LinkingTo:** Rcpp, este campo nos indica si utilizamos scripts de tipo C/C++.
- **Roxygen:** list(markdown = TRUE), esto nos permite una mayor flexibilidad en la documentación del paquete, al permitir el uso de markdown en las anotaciones del Roxygen.
- **RoxygenNote:** Nos indica la versión de Roxygen para la documentación

presente, 7.3.2.

- **Suggest:** Lista de paquetes no esenciales pero sugeridos para funcionalidades adicionales, en este caso testthat(>=3.0.0).
- **Config/testthat/edition:** Configura la edición del testthat utilizada para el testeo, especificando la edición 3.

Este archivo es fundamental a la hora de someter a revisión en el CRAN, pues contiene toda la metadata del paquete, además de facilitar el uso del paquete para los usuarios. Con el archivo `‘‘DESCRIPTION’’` es posible cumplir con los estándares de calidad y normativas del respositorio.

4.3. R

Esta sección constituye a uno de los componentes más cruciales dentro del paquete, puesto que contiene los scripts de `.R`, donde están presente las distintas funciones que se colocan a disposición del usuario y además, contiene funciones que operan de manera interna dentro del paquete, las cuales no son expuesta para el usuario. Estos scripts, debido a la importancia que tienen en la librería fueron detallados en el capítulo anterior, donde se exponen y explican las funcionalidades de cada script presente en el paquete.

4.4. Data

Para el desarrollo de la librería es fundamental que este contenga una estructura de gestión de datos de manera correcta. Para ello, se utiliza la función ***use_ data_ raw(“indipsa”)*** de la librería ***usethis*** (Hadley et al., 2024), la cual facilita la creación de la carpeta *data*. Dentro de esta carpeta se encuentran los scripts diseñados para procesar los datos antes de ser incorporados oficialmente al paquete.
Se crea el script donde preparemos los datos, *prepare_ data.R*, el cual se encargará de leer, procesar y transformar los datos en brutos. Se realiza utilizando el esquema en el código 4.1:

Una vez procesados los datos, es fundamental almacenar los datos en la carpeta *data* del paquete para su acceso y uso. Para esto, se utiliza la misma librería mencionada antes, ***usethis*** (Hadley et al., 2024), se usará la función ***use_ data***

```
indipsa <- read_excel("ruta_de_los_datos")
datos   <- indipsa$CLOSE
indipsa <- ts(datos, start = 2000, freq = 360)
usethis::use_data(indipsa, overwrite = TRUE)
```

Código 4.1: Procesamiento de datos brutos y almacenamiento en formato utilizable.

que guarda los datos en formato *.rda*. En el caso de no existir la carpeta, la función creará la carpeta dentro de la librería automáticamente.

La documentación de los datos incluidos también debe estar incluida en el paquete. Esto se realiza a través de comentarios de **roxygen2** (Wickham et al., 2024). Para nuestro caso, se agrega el script *indipsa.R*, el cual se encuentra en la carpeta **R**, este script se encarga de lo mencionado anteriormente, es decir, de la documentación de los datos incluidos, en el código 4.2 que se encuentra a continuación se exhibe como está documentado.

```
#' Selective Stock Price Index
#'
#' The data covers the period from March 2000 to October 2014,
↪  totaling 3186 observations.
#'
#' @format A time series object with 3186 elements
#' \describe{It's a stock market index that tracks the
↪  performance of a select group of Chilean companies.}
#' @source Yahoo Finance
"indipsa"
```

Código 4.2: Documentación de los datos procesados.

4.5. Test

Para el desarrollo de la librería es fundamental incluir pruebas que verifiquen que las funciones se ejecuten de manera correcta. Para facilitar y ejecutar estas pruebas, se usa la librería ***testthat*** (Wickham, 2011).
Esta carpeta incluye los siguientes archivos:

- **testthat.R** Este archivo actúa como entrada para las pruebas presentes en el paquete. Las pruebas se ejecutan automáticamente cuando se ejecuta

`devtools::test()` (Hadley et al., 2022). La función de esto es configurar el entorno de pruebas y cargar la librería ***testthat*** (Wickham, 2011) junto con el paquete que se va a realizar la prueba. En el código 4.3, se puede ver como está configurado.

```
# This file is part of the standard setup for testthat.
# It is recommended that you do not modify it.
#
# Where should you do additional test configuration?
# Learn more about the roles of various files in:
# *
↪  https://r-pkgs.org/testing-design.html#sec-tests-files-overview
# * https://testthat.r-lib.org/articles/special-files.html

library(testthat)
library(tvGarchKF)

test_check("tvGarchKF")
```

Código 4.3: Script testthat.R

- **test-indipsa.R** Este archivo contiene pruebas específicas para una función, asegurando que se comporte de una manera esperada, según los escenarios. En el código 4.4 se podrán ver las pruebas realizadas.

```
test_that("Valid Input", {
  data(indipsa)
  ipsa<-diff(log(indipsa))
  ipsa<-100*ipsa
  expect_error(tvGarchKalmanFit(ipsa, c = c(0.05,0.05), alpha =
  ↪  c(0.05,0.05), beta = c(0.05,0.05), type =
  ↪  c("trigonometrico","trigonometric","trigonometric"), trig
  ↪  ="cos", arg = "3*(1-log(u))"),"invalid type for c", fixed=T)
  expect_error(tvGarchKalmanFit(ipsa, c = c(0.05,0.05), alpha =
  ↪  c(0.05,0.05), beta = c(0.05,0.05), type =
  ↪  c("trigonometric","trigonometrico","trigonometric"), trig
  ↪  ="cos", arg = "3*(1-log(u))"),"invalid type for a", fixed=T)
  expect_error(tvGarchKalmanFit(ipsa, c = c(0.05,0.05), alpha =
  ↪  c(0.05,0.05), beta = c(0.05,0.05), type =
  ↪  c("trigonometric","trigonometric","trigonometrico"), trig
  ↪  ="cos", arg = "3*(1-log(u))"),"invalid type for b", fixed=T)})

test_that("Los datos están correctamente cargados", {
expect_true(exists("indipsa"))
expect_equal(ncol(indipsa), 1)   # verificar el número de columnas
expect_equal(nrow(indipsa), 3186) # Verificar el número de filas
})
test_that("Example with real data",{
  data(indipsa)
  ipsa<-diff(log(indipsa))
  ipsa<-ipsa[1:3186,1]
  ipsa<-100*ipsa
  fit <- tvGarchKalmanFit(ipsa, c = c(0.05,0.05), alpha =
  ↪  c(0.05,0.05), beta = c(0.05,0.05),type =
  ↪  c("trigonometric","trigonometric","trigonometric"), trig =
  ↪  "cos", arg = "3*(1-log(u))")
  result = c(0.03001, 0.01237, 0.14990, 0.02919, 0.82541, -0.04145)
  expect_equal(fit,result)
})
```

Código 4.4: Script de R para hacer pruebas con las funciones del paquete.

En este caso, nos centramos en los datos de la aplicación, para que nos entregue los resultados esperados, además de arrojar mensajes de error para advertir al usuario de alguna equivocación.

4.6. Man

La carpeta `Man` contiene la documentación de cada script presente en el paquete, la cual está en formato `.Rd` (R documentation). Estos archivos se generan de manera automática a partir de los comentarios especiales en el código `R` mediante la librería `roxygen2` (Wickham et al., 2024). Para generar la documentación es necesario escribir los comentarios especiales directamente en los archivos de código de `R` que existen en el paquete, estos comentarios especiales están precedidos por `#'`, y estos se utilizan para describir funciones, par ámetros, valores de retorno, detalles adicionales y ejemplos.

El código 4.5 a continuación nos muestra una estructura básica de un comentario de `roxygen2` que genera un archivo `.Rd`.

```
#' Titulo de la funcion
#'
#' Descripcion detallada de lo que hace la funcion.
#'
#' @param nombre_param1 Descripcion del primer parametro.
#' @param nombre_param2 Descripcion del segundo parametro.
#' @return Descripcion de lo que devuelve la funcion.
#' @examples
#' # Ejemplo de como usar la funcion:
#' funcion_ejemplo(arg1, arg2)
```

Código 4.5: Estructura de un comentario de roxygen para documentacion.

Adicionalmente, (R Core Team, 2024) tiene un acceso rápido para generar un esqueleto de estos comentarios especiales de la librería `roxygen2`. Al seleccionar lo expuesto en la Figura 4.4, `‘‘Insert Roxygen Skeleton’’` ($Ctrl + Alt + Shift + R$), se obtiene un esqueleto de documentación para la parte superior del script, el cual incluye los parámetros que la función posee.

Al momento de compilar el paquete, se generan los archivos de la documentación `Rd.` correspondientes en la carpeta `man`. Este paso puede ser facilitado por el comando `document()` de la librería `devtools` (Hadley et al., 2022), donde esta función realiza el trabajo de revisar el paquete y generar o actualiza los archivos `Rd.` necesarios. Cada archivo es nombrado como *“nombre-del-script.Rd”* y contiene la documentación estructurada asociada al script con el mismo nombre que está en la carpeta `R`.

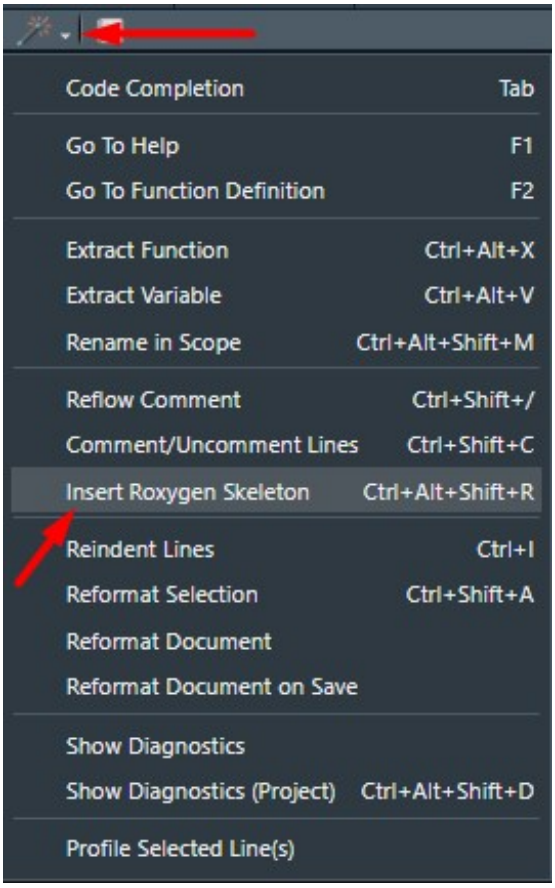

Figura 4.4: Menú en `RStudio` para insertar un esqueleto de la documentación Roxygen.

4.7. Src

La carpeta *src* contiene los archivos `.c` a utilizar en el paquete. Para este caso, seguimos las indicaciones en (R Core Team, 2024), para agregar archivos `.c` al paquete, para esto se incorporan las líneas de código de 4.6 dentro del archivo `.c`.

En este caso, las librerías usadas en `c` nos permite interactuar con el `R`, siendo la primera fundamental para generar la conexión entre los dos lenguajes, el resto de librerías nos sirve para trabajar diferentes operaciones dentro de las funciones. Luego, la función *static const* (R Core Team, 2024) nos permite dejar la función de manera inmutable y además nos ayuda a evitar conflictos con el nombre de la función. La función *R_init_NombreDelPaquete()* (R Core Team, 2024) se utiliza para registrar funciones de `c` para que estén disponibles para el uso en `R` y esto se logra también con la ayuda de *R_registerRoutines()* que nos ayuda a registrar las funciones, según el formato el cual se encuentren, en este caso el formato es *.dll* y

```c
#include <R.h>
#include <Rmath.h>
#include <R_ext/BLAS.h>
#include <Rinternals.h>

// Funciones a usar en R.

static const R_CMethodDef CEntries[] = {
    {"nombre_de_funcion_1", (DL_FUNC)&nombre_de_funcion_1,
     ↪  "cantidad_de_variables_a_la_funcion_1"},
    {"nombre_de_funcion_2", (DL_FUNC)&nombre_de_funcion_2,
     ↪  "cantidad_de_variables_a_la_funcion_2"},
    {NULL, NULL, 0}
};
void R_init_NombreDelPaquete(DllInfo* dll) {
    R_registerRoutines(dll, CEntries, NULL, NULL, NULL);
    R_useDynamicSymbols(dll, FALSE);
}
```

Código 4.6: Código en C para registrar las funciones.

nos indica que la función entrante son provenientes del lenguaje `c` *(CEntries)*. Por último, en el archivo `NAMESPACE` dentro de la librería se registra la función mediante *useDynLib(NombreDelPaquete, .registration = TRUE)*, con esto es posible que los archivos `c` se puedan utilizar en `R`.

4.8. Inst

Esta sección corresponde a citar el paquete. Esta carpeta se genera automáticamente al momento de usar la función `use_citation()` de la librería `usethis` (Hadley et al., 2024), y además se crea un archivo `citation`. Este archivo facilita el proceso de citar el paquete, en el código 4.7 se muestra como citar el paquete `tvGarchKF`.

Esto facilita citar el paquete en referencias de futuros trabajos, esto es, únicamente escribiendo `citation("tvGarchKF")` en la consola de `RStudio` y nos entregará como citar el paquete.

```
> citation("tvGarchKF")
To cite package 'tvGarchKF' in publications use:

  Ferreira G, Arancibia T (2025). tvGarchKF: Estimate
  ↪ parameters of Locally Stationary tvGarch models applying
  ↪ Kalman Filter and
  use Space-State equations_. R package version 0.0.1.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {tvGarchKF: Estimate parameters of Locally
    ↪ Stationary tvGarch models applying Kalman Filter and
    ↪ use Space-State equations},
    author = {Guillermo Ferreira and Tomás Arancibia},
    year = {2025},
    note = {R package version 0.0.1},
  }
```

Código 4.7: Citar el paquete en referencias.

4.9. Instalación

Como se mencionó al comienzo del capítulo, el paquete esta en el repositorio de `Github`, donde se encuentran sus archivos correspondientes:

https://github.com/Tomas-UDEC/tvGarchKF

Además, si se ejecuta la función `install_github` de la librería `devtools` (Hadley et al., 2022) se podrá instalar el paquete de manera sencilla y rápida, siempre y cuando se tenga acceso a internet. Como se muestra en el código 4.8, en la consola de R, se podrá instalar de manera sencilla el paquete:

```
devtools::install_github("Tomas-UDEC/tvGarchKF")
```

Código 4.8: Instalar paquete `tvGarchKF` desde Github a través de la consola.

5. Conclusión y Trabajos futuros

5.1. Conclusión

En esta tesis se desarrolló una librería **tvGarchKF** para el software *R* y se implementó para simulaciones y aplicación de datos reales del *IPSA* para modelar la volatilidad financiera con características no-estacionarias. La implementación ha generado buenos resultados en varios escenarios de volatilidad.

Durante el desarrollo de la librería se logra observar que la opción de implementar dos lenguajes resulta más eficiente, en este caso, lenguaje *C* y *R*, donde ambos lenguajes son igual de importantes. Debido a la incorporación del lenguaje *C* a la librería se mejora la eficiencia computacional. Además, el estudio de simulación realizado por repeticiones Monte Carlo presentó buenos resultados para las estimaciones del modelo `tv-Garch`.

Esta librería nos entrega una manera eficaz de trabajar modelos `tv-Garch` a través de procedimientos con la metodología `KF`, trabajando con ecuaciones espacio-estado, además para analizar se trabaja con simulaciones Monte Carlo para estimar los parámetros *time-varying*. Esta tesis se enfocó únicamente en modelos `tv-Garch`(1,1), por lo que existe la posibilidad de generar extensiones para modelos `tv-Garch`(p,q), lo cual queda sujeto a futuras investigaciones. Por lo que la librería esta hecha para modelos `tv-Garch`(1,1).

El paquete ofrece herramientas para desarrollar modelos `tv-Garch`(1,1) para modelos lineales, no lineales y trigonométricos, lo cual nos permite generar un análisis en torno a estos modelos, como por ejemplo, generar estimaciones a los parámetros *time-varying*, predicciones. Expandir más las estructuras de los modelos es parte de los trabajos futuros.

5.2. Trabajos futuros

- Ampliar el espectro de valores para p y q en los modelos `tv-Garch`.
- Mejorar el input de las funciones c, α y β.
- Exponer las funciones internas a la comunidad.
- Mantención y actualización de la librería.

Referencias

Amado, C. and Terasvirta., T. (2008). Modelling conditional and unconditional heteroskedasticity with smoothly time-vaying structure. *SSE/EFI Working paper Series in Economics and Finance, (691).*

Bollerslev, T. (1986). Generalized autoregressive conditional heterocedasticity. *Journal of Econometrics , 31:307–327.*

Brockwell, P. J. and Davis, R. A. (2002). *Introduction to time series and forecasting.* Springer.

Carnero, M. A., Peña, D., and Ruiz, E. (2012). Estimating garch volatility in the presence of outliers. *Economics Letters*, 114(1):86–90.

de Jong Piet and Jeremy, P. (2004). The arma model in state space form. *Statistics & probability letters*, 70(1):119–125.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica: Journal of the econometric society*, pages 987–1007.

Ferreira, G., Navarrete, J. P., Rodríguez-Cortés, F. J., and Mateu, J. (2017). Estimation and prediction of time-varying garch models through a state-space representation: a computational approach. *Journal of Statistical Computation and Simulation*, 87(12):2430–2449.

Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222(594-604):309–368.

Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154.

Hadley, W., Jennifer, B., Malcolm, B., and Andy, T. (2024). usethis: Automate package and project setup. r package version 2.2. 3.

Hadley, W., Jim, H., Winston, C., and Bryan, J. (2022). devtools: tools to make developing r packages easier. r package version 2.4. 5. *URL https://CRAN. R-project. org/package= devtools.*

Lorenzo, P., Juan, R., and Esther, R. (2006). Bootstrap prediction for returns

and volatilities in garch models. *Computational Statistics Data Analysis*, 50(9):2293–2312.

Nelson, D. B. and Cao, C. Q. (1992). Inequality constraints in the univariate garch model. *Journal of Business & Economic Statistics*, 10(2):229–235.

Nocedal, J. and Wright, S. J. (2006). Quasi-newton methods. *Numerical optimization*, pages 135–163.

Patilea, V. and Raïssi, H. (2014). Testing second-order dynamics for autoregressive processes in presence of time-varying variance. *Journal of the American Statistical Association*, 109(507):1099–1111.

Penzer, J. (2007). Inference for garch and related models with incomplete data. *Department of Statistics*.

R Core Team (2024). *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria.

Rainer, D. (1997). Fitting time series models to nonstationary processes. *The annals of Statistics*, 25(1):1–37.

Rainer, D. and Suhasini, S. R. (2006). Statistical inference for time-varying arch processes.

Rohan, N. (2013). A time varying garch (p, q) model and related statistical inference. *Statistics & Probability Letters*, 83(9):1983–1990.

Rohan, N. and Ramanathan, T. (2013). Nonparametric estimation of a time-varying garch model. *Journal of Nonparametric Statistics*, 25(1):33–52.

Wickham, H. (2011). testthat: Get Started with Testing. *The R Journal*, 3(1):5–10.

Wickham, H., Danenberg, P., Csárdi, G., and Eugster, M. (2024). roxygen2: In-line documentation for r (r package version 7.2. 3).

A. Apéndice

A.1. Códigos en R

A continuación se presentan los parámetros utilizados por las funciones de script en R junto a sus descripciones. Los parámetros usados en las funciones de R son:

- **n** (*numeric*): Es el valor del largo de la serie que se va a simular.
- **x** (*numeric vector*): Son los coeficientes a ajustar en la función *tvGarchKalmanFit*.
- **Series** *(data.frame)*: Es una lista o entorno que contiene los datos de la serie a utilizar para el modelo.
- **gamma** (*numeric vector*): Son los coeficientes para la función $c(u)$ en el modelo.
- **c** (*numeric vector*): Son los coeficientes para la función $c(u)$ en el modelo.
- **alpha** (*numeric vector*): Son los coeficientes para la función $\alpha(u)$ en el modelo.
- **beta** (*numeric vector*): Son los coeficientes para la función $\beta(u)$ en el modelo.
- **nsample** (*numeric*): Es el largo de los datos ingresados.
- **type** (*string vector*): Es un vector el cual contiene el tipo de función que corresponde para $c(u), \alpha(u), \beta(u)$.
- **exponentes** (*numeric vector*): Son los exponentes para el caso *NoLineal*.
- **trig** (*string*): Parámetro que nos indica la función trigonométrica a usar, *cos* o *sin*.
- **arg** (*string*): Es el valor del argumento a usar en la función trigonométrica.
- **predict** (*numeric*): Es la variable que indica la cantidad de tiempo para predecir.

A.2. Códigos en C

A continuación se muestra el código en C utilizado para el desarrollo del paquete. Los parámetros usados para `tvGarchKF.c` son:

- **nsample**: Esta variable corresponde al tamaño de muestra.
- **a**: Son los coeficientes de $\alpha(u)$.
- **b**: Son los coeficientes de $\beta(u)$.
- **c**: Son los coeficientes de $c(u)$.
- **P**: Son los valores para $\Omega_{t+1,T}$ en las ecuaciones (2.9).
- **X**: Son los valores para $\widehat{\xi}_{t+1,T}$ en las ecuaciones (2.9).
- **F.m**: Esta variable corresponde a la varianza en las ecuaciones (2.9).
- **K**: Son parte del desarrollo para ambas ecuaciones (2.9), en este caso, esta variable es igual a $\Theta_{t,T}$ en las ecuaciones (2.9).
- **y**: Sea *Serie* los datos ingresados de la serie de tiempo. Entonces, la variable y corresponde a $Serie^2 - c(u)/(1 - \alpha(u) - \beta(u))$.