GENERACIÓN DE PRONÓSTICOS DE UN SISTEMA EN TIEMPO REAL USANDO R

Trabajo de Grado Presentado Para Obtener el Título De

Especialista en Estadística Aplicada

Ingeniero Electrónico

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES

FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS

ESPECIALIZACIÓN EN ESTADÍSTICA APLICADA

BOGOTÁ D.C.

2017

GENERACIÓN DE PRONÓSTICOS DE UN SISTEMA EN TIEMPO REAL USANDO R

JUAN PABLO QUINTERO CASTRILLÓN

Ingeniero Electrónico

ASESOR: MSc. JUAN CAMILO SANTANA

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES

FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS

ESPECIALIZACIÓN EN ESTADÍSTICA APLICADA

BOGOTÁ D.C.

2017

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del Jurado

_____

Firma del Jurado

Bogotá, D.C. 11 de diciembre del 2017

Agradecimientos

El autor expresa sus agradecimientos a:

A mi familia; ya que, sin su paciencia y comprensión no se hubiera podido finalizar este trabajo.

A los docentes de la Fundación Universitaria Los Libertadores, por su constante acompañamiento y sus valiosos aportes en conocimientos a lo largo de la especialización.

Tabla de contenido

Listado de Ilustraciones

Listado de Ecuaciones

Página.

Listado de Tablas

Generación de pronósticos de un sistema en tiempo real usando r

JUAN PABLO QUINTERO CASTRILLÓN

Capítulo 1. Introducción

En las organizaciones, siempre hay una relación entre las directrices y los resultados obtenidos, dentro de este sistema se encuentran los procesos productivos, los cuales reciben la materia prima y por diferentes medios (Químicos, físicos, mecánicos y mixtos) se obtiene un producto terminado.

Esto combinado al nivel de tecnificación en la toma de información relevante en las organizaciones, produce un retardo entre: La recolección de la información, el filtrado, el análisis y las conclusiones para la toma de decisiones importantes.

Estos tiempos que pueden variar entre horas, días o semanas, afectan de manera importante el progreso de cada organización; ya que, al momento de la toma de decisiones pueden presentarse cambios que lleven a perder vigencia y efectividad en el desarrollo de su actividad económica.

Objetivo General

Diseñar una metodología de análisis estadístico basado en series temporales para el análisis de bases de datos de origen externo que permita la generación de pronósticos en tiempo real.

Objetivos Específicos

- Crear mecanismos de intercomunicación entre R y un motor de base de datos SQL.
- Filtrar automáticamente y ajustada en el tiempo, variables críticas de análisis.
- Definir un procedimiento por código en R que permita tomar las informaciones desde una base de datos SQL.
- Generar un pronóstico con base en la metodología de series de tiempo.
- Retroalimentar la base de datos con los datos de pronóstico generados desde el código en R.

Capítulo 2. Marco teórico

En los sistemas de producción es en donde están las tareas más críticas, las cuales son divididas en varias etapas, y es allí donde dependiendo de su tecnología y/o personal, el proceso puede mantenerse en control, o sí, por el contrario, alguna de estas etapas falla, el sistema completo puede llegar incluso a colapsar, lo cual en una organización representa pérdidas económicas, incumplimientos, sobrecostos, y penalizaciones.

Todo lo anterior, es una pequeña representación de la eterna diferencia entre lo que buscan los cargos administrativos, comparado con los resultados obtenidos por las áreas operativas. En los casos donde la información del resultado se tarda mucho tiempo en llegar a las directivas, las decisiones que se toman pueden o no ser acertadas; ya que, no se toman en el momento justo y esta brecha se repite una y otra vez.

Con el avance de los sistemas electrónicos y su conectividad, es posible ahora el uso de tecnologías para diseñar un procedimiento que involucre los temas de competencias técnicas; todo esto permite llevar a cabo la adquisición automática de los parámetros, su organización en bases de datos y finalmente el procedimiento estadístico óptimo, con el fin de realizar el análisis de dicha información y obtener los resultados mencionados.

Así pues, la disponibilidad de la información en “tiempo real” sobre el proceso, ayudaría a tomar decisiones acertadas, evitando que se salga de control y actuar de manera oportuna.

A continuación, están los temas relacionados con el desarrollo del trabajo, tanto en conceptos técnicos como tecnológicos.

2.1 Tiempo Real

En la actualidad, no hay una sola definición para este término, debido a que depende exclusivamente del argot técnico en el cual se quiere definir. Como unidad de medida, el tiempo nos permite enmarcar eventos dentro de una secuencia lógica, y es una parte fundamental para entender el mundo que nos rodea.

Sin la enmarcación de los eventos en una secuencia, sería imposible para nosotros comprender la realidad de lo que sucede a nuestro alrededor, por ejemplo: si empujamos un vaso de una mesa, esperamos que después de cierto tiempo (evento que se puede calcular) el vaso caerá al suelo y si este es de un material frágil se romperá; lo que nunca esperamos en la realidad es que suceda al revés.

En los tiempos modernos, enmarcamos muchos de los eventos usando computadores y software, que permiten almacenarlos de manera ordenada, en un tiempo definido, y con la gran ventaja que podemos realizar consultas a gran velocidad.

El detalle en este punto es que los computadores, en su funcionamiento tienen un tiempo de ciclo, que depende propiamente de la electrónica concentrada en su procesador.

Surge de este tipo de aplicaciones, la necesidad de definir un marco temporal, que permita relacionar los eventos almacenados o procesados en un computador, con el

mismo tiempo que usamos como humanos. Vinculado el tiempo real, se logra que un evento que sucede en una fecha de calendario, a una hora del día, quede registrado en un lenguaje entendible tanto para la máquina (computador) como para el humano y esto abre el espacio al concepto de tiempo real.

Para nuestro caso, el tiempo real, se usa como herramienta para medir la cercanía que tiene la toma de los datos, el análisis, la generación de pronóstico y su enmarcación con las unidades de tiempo que usamos en la vida cotidiana, usando algoritmos programados en R y su iteración con el ambiente de Windows.

2.2 Automatización de Proceso

En los últimos años, se han venido dando implementaciones tecnológicas importantes, que han permitido realizar tareas de manera automática donde antes se tenían que hacer de una manera manual y rutinaria, un ejemplo de esto son las máquinas expendedoras de alimentos, sistemas de cobros, etc.; los cuales eran antes realizados por personas, y que han sido reemplazados por sistemas de diferentes tipos de complejidad.

Con el ejemplo anterior, podemos observar que los trabajos que tienden a ser más automatizados son aquellos que tienen una rutina definida, y que, con el paso del tiempo, éstas cambian poco o nada, y que tecnológicamente en los mercados, existen herramientas para llevar a cabo estos tipos de procesos.

Definimos entonces la automatización de procesos, como el conjunto de técnicas y tecnologías que permiten realizar tareas repetitivas de una manera automática.

2.3 Bases de Datos

En la medida que los procesos son más complejos y mejor controlados, se hace necesario recolectar y almacenar mucha información; y a medida que ésta crece, se convierte en todo un reto administrarla, tanto para el almacenamiento efectivo como en las capacidades de los equipos que logran hacerlo.

Para hacernos una idea, imaginemos tener un directorio telefónico en donde no solo podemos consultar cada nombre y el teléfono, sino, además: Dirección, edad, medidas corporales, estado civil, etc.

Esta información debe estar relacionada entre sí, por medio de llaves o relaciones que permitan encontrarla de manera sencilla y rápida, evitando en lo posible duplicar información, para que su tamaño en el computador sea el mínimo posible.

Como base de datos, también debe permitir realizar consultas dentro de la misma, y arrojar información según criterios ingresados, de una manera intuitiva y con mínimos conocimientos de programación.

Para el caso de este trabajo, la base de datos se crea en un motor llamado SQLEXPRESS, el cual es tipo SQL (Estructured Query Language) o lenguaje de consulta estructurada, que permite realizar diferentes operaciones, pero principalmente tres tipos: una de consulta, una de borrado y otro de escritura de datos dentro de esta base de datos.

Las bases de datos constan de diferentes partes que los componen. Estos elementos de manera jerárquica son los siguientes:

a. **Motor de base de datos**: Servicio dentro del ambiente Windows, que permite mediante instrucciones básicas como consultar, introducir o modificar; información dentro de una base de datos creada dentro de sí.
b. **Base de datos**: Es la ubicación lógica, dentro del motor de base de datos, que contienen la información organizada en manera de tablas. Tiene un nombre único dentro del motor de base de datos, y se crea a partir de reglas.
    a. Tablas: Es el espacio lógico en el que se diseña la estructura de cómo se almacena la información, cada tabla, debe ser asociada a una categoría de información, para que la base de datos en su totalidad sea organizada y más fácil de consultar. Ejemplo: dentro de la base de datos A, puede existir una tabla para la información del cliente (nombre, identificación, contacto, email) y otra de ventas, que puede estar relacionado con la primera de cliente (ventas por cliente, fechas, monto, sistema de pago)

        i. Columnas: Dentro de una tabla corresponden a los encabezados de la información, que se va a ir almacenando, por ejemplo, la columna “nombre” contendrá todos los nombres de cada uno de los registros en la tabla, otra columna de “número telefónico” tendrá todos los números telefónicos de los registros dentro de la tabla formulario
        ii. Registros: Cada ingreso de información que corresponde a las categorías definidas por las columnas son un registro, cada registro debe cumplir con las reglas que se definen en las

columnas, como tipo de datos, y su exigencia de estar diligenciada o no.

En el caso del proyecto actual, la base de datos ya está creada previamente por un aplicativo denominado “software de adquisición de datos”, que va introduciendo registros en las tablas, en una unidad de tiempo definida.

2.3 Software de Adquisición de Datos

En las aplicaciones de la vida real siempre generamos información, la cual dependiendo del proceso puede ser leída de manera electrónica; ya sea, por un dispositivo de medición electrónico o como consecuencia de un registro digital realizado por un usuario o la selección de opciones dentro de un software especial.

Cuando esta información es tomada de manera electrónica, por medio de herramientas del software y del hardware se pueda realizar un muestreo cada cierto tiempo y almacenar esta información en una base de datos.

Los paquetes de software que se encargan de realizar estas tareas son denominados “software de adquisición de datos”, el cual contempla varias etapas, las cuales se muestran a continuación:

a. **Dispositivo electrónico para tomar la información**: Se conoce como sensor, mide la variable física, el evento o el proceso y la convierte en una señal eléctrica que puede ser interpretada por un sistema de cómputo.

b. **Sistema de comunicación**: Son aquellos que permiten enviar y/o recibir información entre dos puntos, los cuales para el alcance del proyecto actual serían desde el dispositivo electrónico que toma la información y el sistema de cómputo.
c. **Software de adquisición**: Es un programa encargado de gestionar la comunicación con los dispositivos que capturan la información, mediante la conexión con la base de datos. Logrando tomar la información del proceso, organizarla y adecuarla según las exigencias.

En el alcance del proyecto, el software es suministrado por la empresa **SMARTPRO SAS**, la cual realizo el desarrollo intelectual, siendo este uno de los servicios que presta en su portafolio.

2.4 Modelo ARIMA Para Series de Tiempo Estacionarias

Las series de tiempo, en su concepto, son colecciones de observaciones hechas secuencialmente en el tiempo (1), al graficar las series de tiempo, podemos ver el comportamiento de una variable en el tiempo, pudiendo inferir si muestra tendencias o ciclos.

Para las series de tiempo, existe un método de análisis que tienen en cuenta la información de una sola variable; que suponemos, contiene en ella misma la información necesaria, y sus posibles relaciones directamente en su resultado, por lo que, según la metodología, podemos realizar un modelo que permite el pronóstico en el futuro.

Para poder analizar la información se debe conocer los conceptos básicos para lograr una evaluación estadística valida, estos conceptos son los siguientes:

a. Estacionaridad: Cuando las funciones de distribución conjuntas son invariantes con respecto a un desplazamiento en el tiempo.
b. Tendencia: Cuando la serie de tiempo, exhibe un cambio de media con el paso del tiempo, se dice que la serie de tiempo tienen tendencia, lo cual indica que el valor esperado va aumentando, aunque esto va a depender directamente de la cantidad de datos tomados; ya que, en series de tiempo con muchos datos, se puede empezar a observar el fenómeno de estacionalidad, ejemplo: las ventas de una empresa en un periodo de varios años, puede mostrar una tendencia a la alza, lo cual significaría, que en los últimos años el promedio de ventas es mayor al inicio de la medición.
c. Estacionalidad: Muchas series de tiempo muestran un comportamiento cíclico dentro de un periodo de tiempo, lo que implica que existe un patrón repetitivo en la forma que se genera la información, por ejemplo: la temperatura de una ciudad con estaciones, de un año a otro muestra de una manera muy marcada cuándo es el invierno y cuando es el verano, repitiendo este patrón en los siguientes años.

Para la realización del análisis de la serie de tiempo, según la metodología Box-Jenkins, se debe tener una serie estacionaria; lo cual, en términos prácticos, significa que

no hay cambio sistemático en la media (no hay tendencia), si no hay cambio en la varianza, y si las variaciones periódicas han sido removidas.

La primera técnica que se aplica para volver la serie estacionaria es la diferenciación, la cual es un tipo de filtro para la información, el cual consiste en la diferencia de cada valor, por su rezago inmediatamente anterior.

Ecuación 1 $Y_t = X_{t+1} - X_t$

En el caso del presente trabajo, la diferenciación, se usa cada vez que se determina que la serie de tiempo no es estacionaria. La prueba estadística que se utiliza es la Dickey Fuller.

Esta prueba, determina si en la serie de tiempo, existe o no, raíces unitarias, para el caso del test, la hipótesis nula, es la existencia de raíces unitarias en la serie.

Teniendo en cuenta lo anterior, el modelo ARIMA, realiza un modelo de la serie de tiempo, buscando los factores AR (autorregresivos), que explican cómo afecta los choques de los rezagos anteriores a la información actual; y los factores MA (Media móvil), la cual indica que el error de la regresión es una combinación lineal de los términos de los errores. Por lo anterior el modelo ARIMA, busca los factores φ(autorregresivos) y θ (media móvil) para cada rezago, y los cuales pueden explicar los valores actuales de la serie de tiempo.

Ecuación 2. Modelo AR(p) $X_t = c + \sum_{i=1}^{p} \varphi_i * X_{t-i} + \varepsilon t$

Ecuación 3. Modelo MA(q) $X_t = \mu + \varepsilon t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-1}$

Ecuación 4. Modelo ARMA(p,q) $X_t = c + \varepsilon t + \sum_{i=1}^{p} \varphi_i * X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-1}$

Mediante el ajuste de estos modelos, podemos generar una descripción de los valores de la serie de tiempo, que dependen de la información dentro de esta, logrando así, usar la información que la serie de tiempo tienen integrada, y logrando una regresión basada en una sola variable; aunque con el riesgo que implica, al mismo tiempo, no tener en cuenta todas las interacciones que tiene la serie con otras variables.

Por último la validez del modelo se hace por medio de dos test estadísticos que son:

- **Test Ljung-Box**: El test evalúa si un grupo cualquiera de correlaciones de una serie de tiempo son diferentes de cero, verifica que la serie es aleatoria basado en un número de retardos o lags, determinando si la variabilidad de la serie puede explicar la información, lo que se busca en rechazar la hipótesis nula, que estadísticamente aporta la evidencia que la serie es independiente, por lo tanto, los valores son aleatorios.
- **Test Jarque-Bera**: Esta prueba determina la asimetría y la curtosis de una muestra de datos, con el fin de determinar si estas tienen una tendencia de

normalidad, lo cual aporta evidencia estadística, que dicha información es fiable, y los pronósticos son en un porcentaje definido, confiables.

Capítulo 3. Marco Metodológico

Los datos usados para la evaluación del procedimiento son basados en los comportamientos de producción semana a semana, que pueden estar en una empresa de producción, estos datos son generados con base en la experiencia del autor del trabajo, y están divididas en semanas.

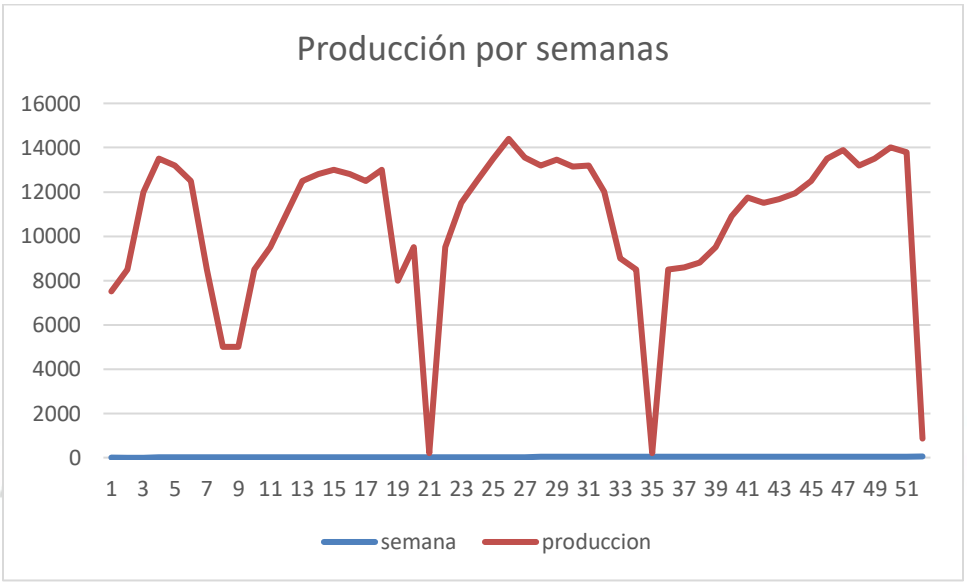


Ilustración 1 Producción en una empresa por semanas

La gráfica muestra un comportamiento de producción típico, con puntos altos y bajos, los cuales son resultados del desempeño de las máquinas y de los procesos productivos, este tipo de gráficas están marcadas por un techo, el cual sería la producción ideal teniendo en cuenta que las máquinas están trabajando constantemente y al momento de una falla en estas, también lo hace el proceso, ya sea por errores de

operación o bajas ventas, la producción tiene caídas que pueden llegar a cero, lo cual en este caso hipotético podría representar grandes pérdidas económicas.

El trabajo presentado busca determinar mediante un algoritmo en el software R, los puntos críticos de un proceso sirviendo de alarma temprana o advertencia para que estos momentos sean afrontados con mayor preparación.

Para lograr el objetivo general del proyecto, se realiza una división de tareas, con base en un diagrama de flujo planteado, para llegar al procedimiento general el cual busca el pronóstico de una serie de tiempo, cuya información se encuentra en una base de datos tipo SQL, y que, para su análisis estadístico se va a usar el software de programación enfocado en la estadística R. A continuación, se indica el diagrama de flujo del proceso planteado.

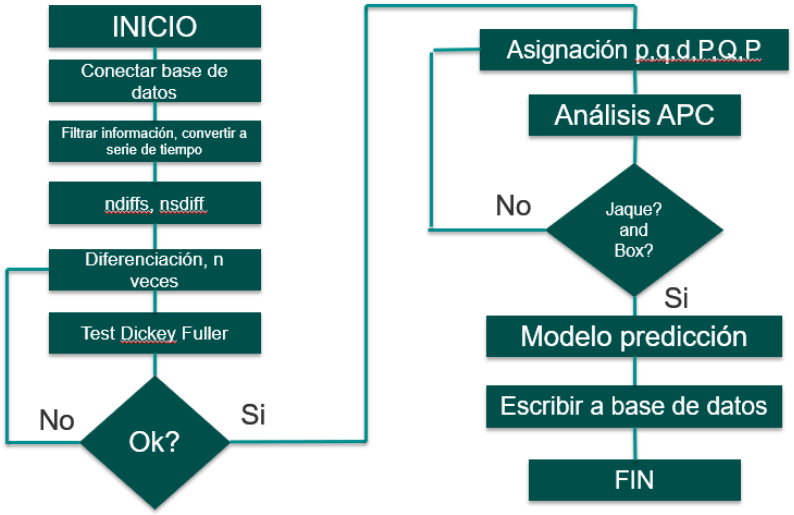


Ilustración 2. Diagrama de flujo del procedimiento

3.1 Conexión Entre R y Una Base de Datos SQL

En el primer paso del diagrama de flujo, la conexión de R con la base de datos origen, se hace mediante comandos en R especiales que se encuentran en la librería RODBC, la cual permite realizar una conexión con el motor de base de datos, y por medio de instrucciones SQL (Lenguaje de Consulta Estructurada), realizar consultas con una semántica que asemeja la forma con que los humanos lo hacemos en nuestro día a día.

Para realizar este tipo de operaciones, se debe tener conocimiento previo en el manejo de base de datos tipo SQL y de sus respectivos motores, además de tener cierto manejo en configuraciones avanzadas en el entorno Windows.

Los pasos para lograr esta conectividad se describen a continuación.

- Configuración del DSN (Data Source Name): El DSN, permite que Windows publique dentro de su entorno, un nombre relacionado con una base de datos, reduciendo los parámetros de configuración para las demás aplicaciones y simplificando el proceso de conexión; ya que, solo requerirá un nombre definido en el procedimiento. Este proceso se muestra a continuación.
    - **Abrir las herramientas administrativas de Windows.**

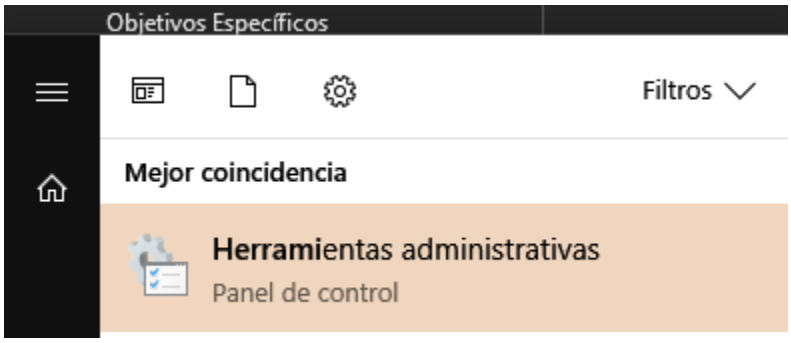


Ilustración 3. Opción de herramientas administrativas mostrado en el menú inicio.

- **Seleccionar fuentes ODBC 32 bits.**

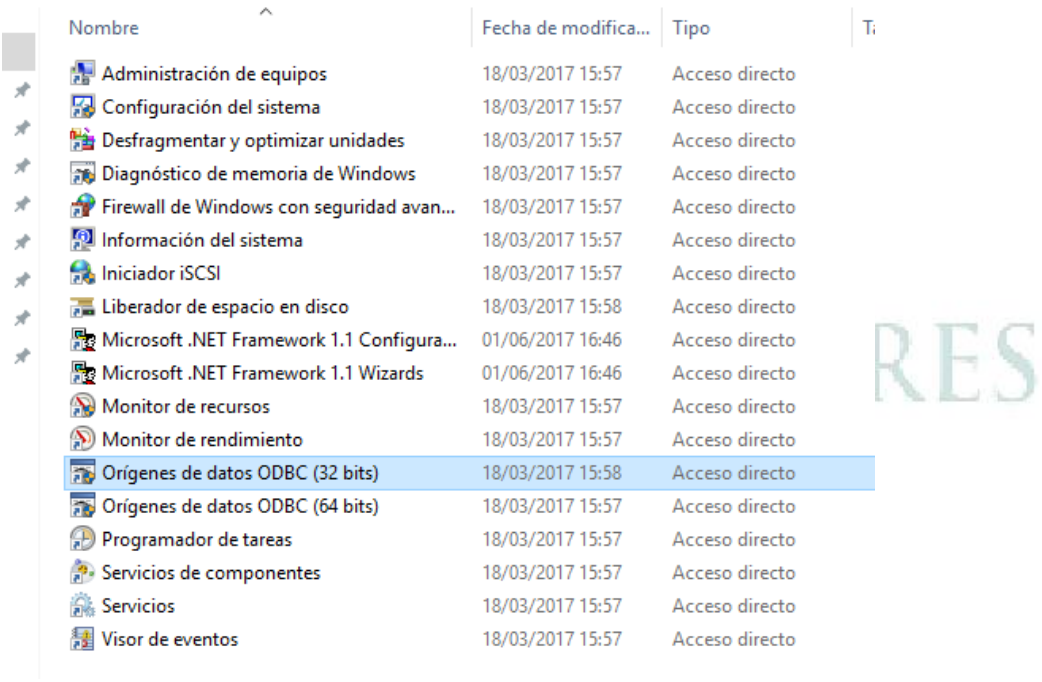


Ilustración 4. Opción de orígenes de datos ODBC 32 bits

- **En la ventana de orígenes de datos, se da clic en agregar.**

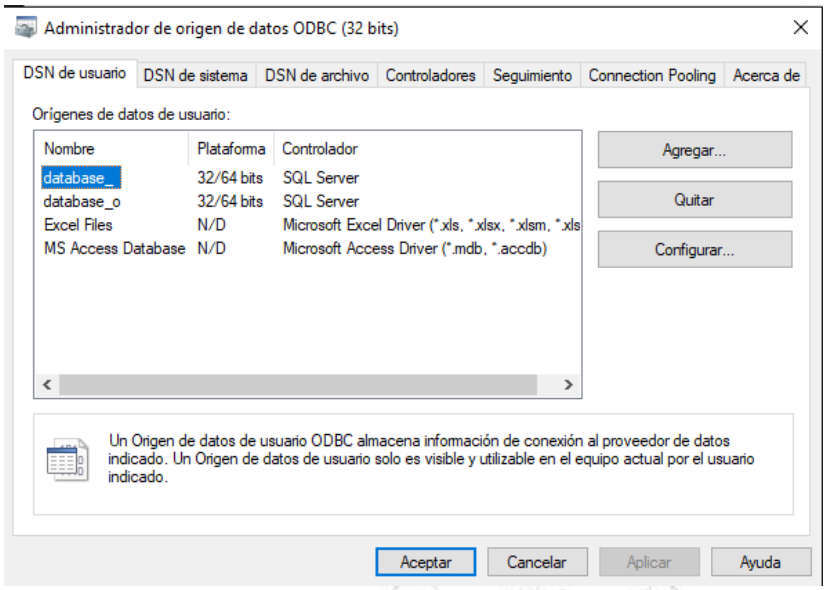


Ilustración 5. Ventana del administrador de datos ODBC.

- **Se debe seleccionar tipo de servidor SQL, para el caso en el que el motor de base de datos sea SQLEXPRESS**

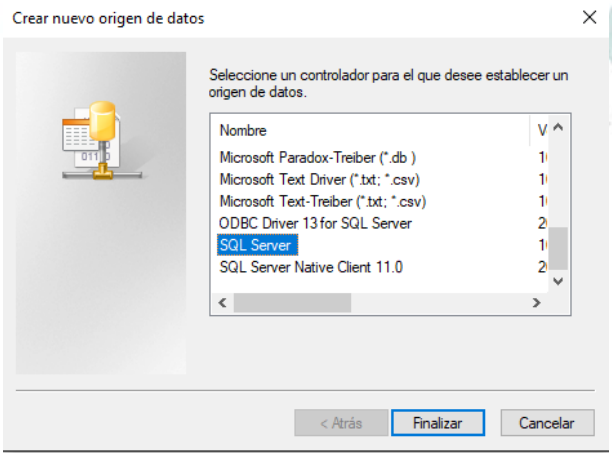


Ilustración 6. Selección del tipo de motor de base de datos.

- **Ingresar el nombre, descripción y la ruta donde se encuentra el motor de base de datos. El nombre que se diligencia en este paso es**

con el que queda identificado el origen de datos dentro de Windows.

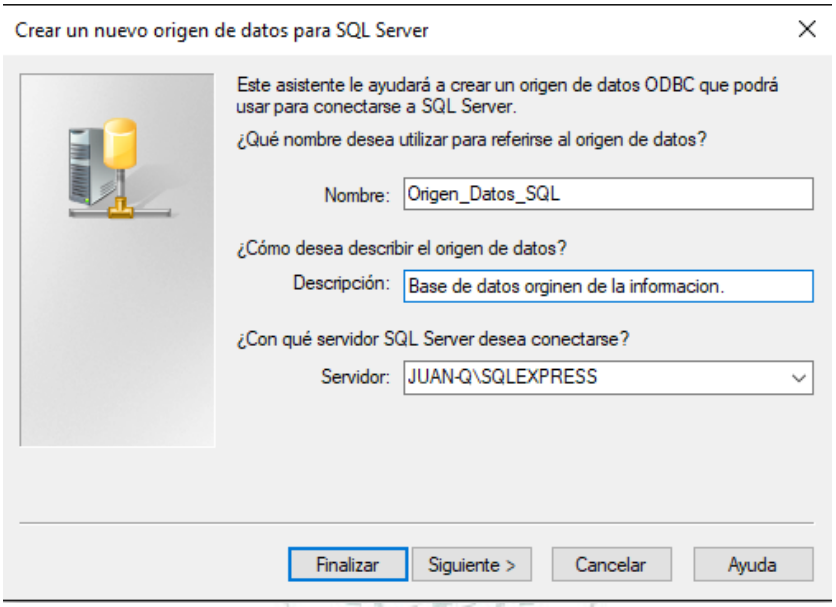


Ilustración 7. Formulario con las configuraciones iniciales de la conexión.

- **La autenticación, debe corresponder a la que tiene la base de datos, la más usada es la autenticación con contraseña del inicio de sesión de Windows**

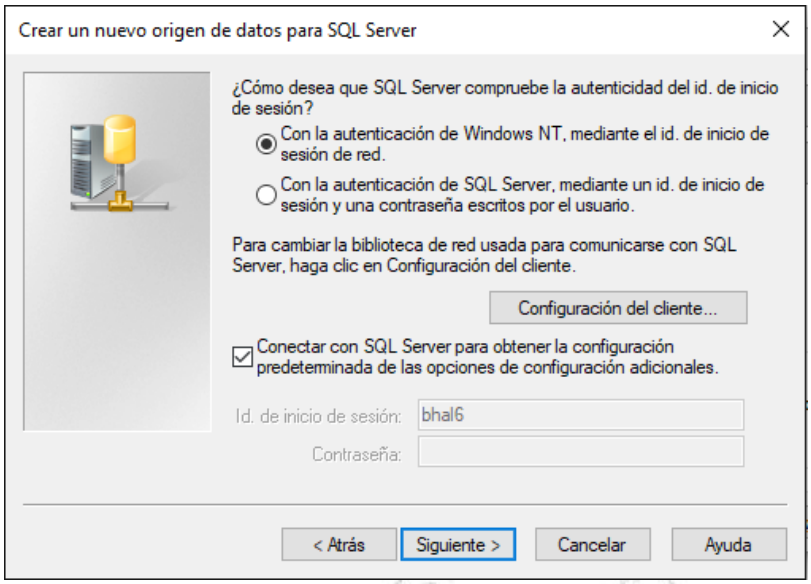


Ilustración 8. Verificación del tipo de autenticación.

- **Se debe seleccionar la base de datos que contiene la información dentro del motor de base de datos**

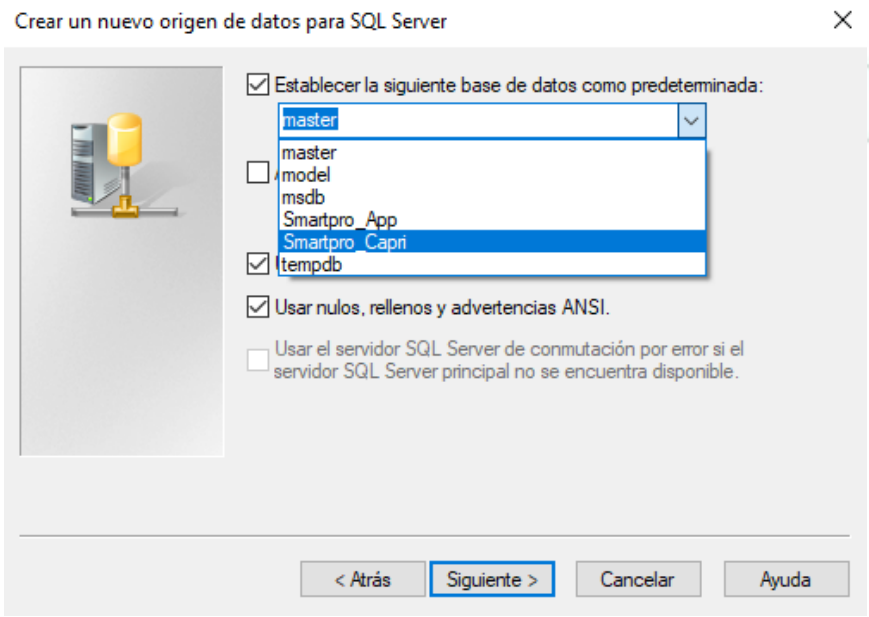


Ilustración 9. Selección de la base de datos con la información requerida.

- **En el último paso, no se realizan cambios, a no ser, que se requiera funcionalidades adicionales que no están dentro del alcance de este proyecto.**

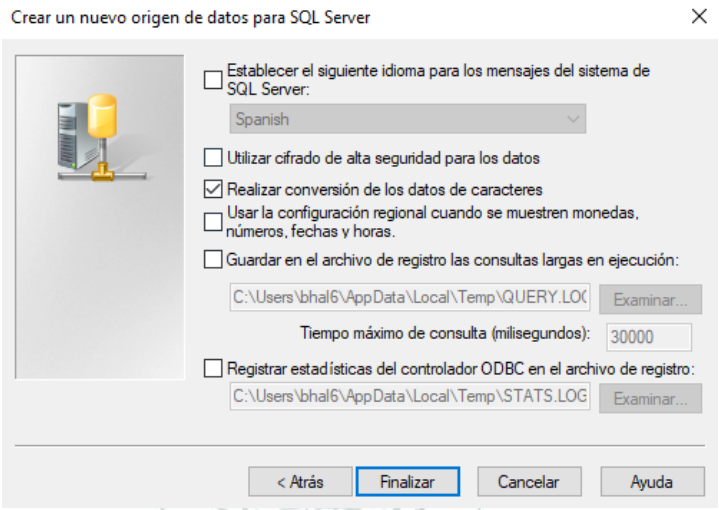


Ilustración 10. Opción de configuraciones adicionales.

- **Finalizar, y probar la conectividad**

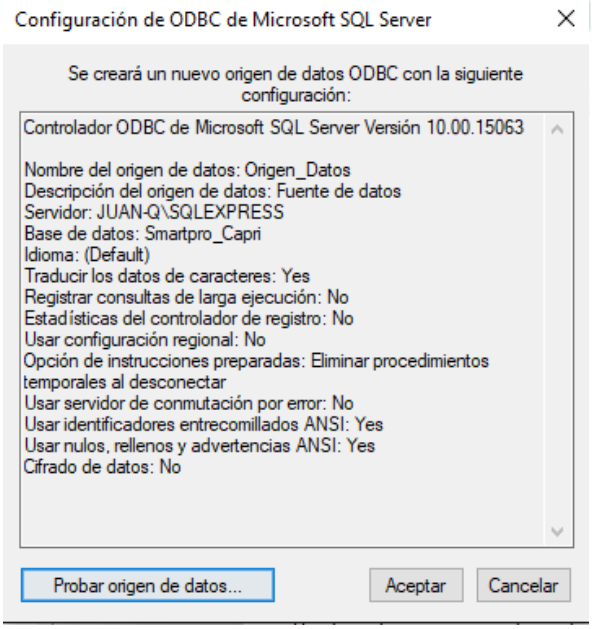


Ilustración 11. Ventana para prueba de conectividad y salida de resultados.

- **Código de R, para la conexión con el origen de la base de datos:** Ya con la configuración de DSN, podemos proceder el código que se encarga de realizar la conexión entre R, y las tablas de la base de datos que contienen la información. La librería usada para este objetivo es la RODBC, que se puede descargar con la siguiente línea de código

  install.packages("RODBC")

  esta línea instala el paquete con las instrucciones que se requieren para la conexión de las bases de datos. Seguido de este paso se ejecutan las siguientes líneas de código.

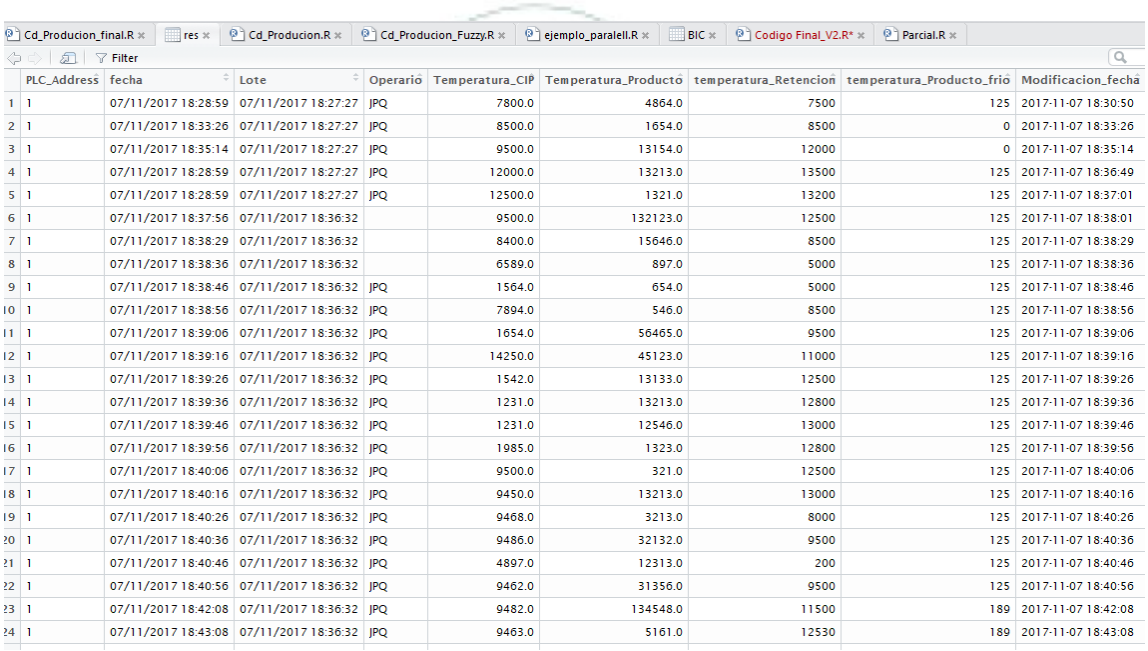
```
dbhandle <- odbcConnect("Origen_Datos")
res <- sqlQuery(dbhandle, 'SELECT TOP(100) * FROM temperaturas')
res
```

Ilustración 12. Código para la conexión e importación de datos desde una tabla.

La variable *dbhandle*, contiene la información necesaria de la conexión de R con el motor de base de datos y la base de datos como tal, y que devuelve la función *odbcConnect*, este identificador, se usa para todas las instrucciones que tienen relación con las consultas de la información de la base de datos. Con la instrucción sqlQuery(), que tiene como argumento el identificador de la conexión o la variable *dbhandle* y la cadena de caracteres con la instrucción de consulta, que retorna dicha información a una variable denominada *res*.

La cadena “SELECT TOP(100) * FROM temperaturas”, usa la instrucción SELECT para indicar que se van a pedir los registros que cumplan con las condiciones que lo prosiguen, TOP(100) indica que los registros solicitados son los primero 100 de la tabla temperaturas, en donde se encuentra la información de interés.

A continuación, se muestra el resultado *res*:



Cd_Producion_final.R | res | Cd_Producion.R | Cd_Producion_Fuzzy.R | ejemplo_paralell.R | BIC | Codigo Final_V2.R* | Parcial.R

Filter

	PLC_Address	fecha	Lote	Operario	Temperatura_CIP	Temperatura_Producto	temperatura_Retencion	temperatura_Producto_frio	Modificacion_fecha
1	1	07/11/2017 18:28:59	07/11/2017 18:27:27	JPQ	7800.0	4864.0	7500	125	2017-11-07 18:30:50
2	1	07/11/2017 18:33:26	07/11/2017 18:27:27	JPQ	8500.0	1654.0	8500	0	2017-11-07 18:33:26
3	1	07/11/2017 18:35:14	07/11/2017 18:27:27	JPQ	9500.0	13154.0	12000	0	2017-11-07 18:35:14
4	1	07/11/2017 18:28:59	07/11/2017 18:27:27	JPQ	12000.0	13213.0	13500	125	2017-11-07 18:36:49
5	1	07/11/2017 18:28:59	07/11/2017 18:27:27	JPQ	12500.0	1321.0	13200	125	2017-11-07 18:37:01
6	1	07/11/2017 18:37:56	07/11/2017 18:36:32		9500.0	132123.0	12500	125	2017-11-07 18:38:01
7	1	07/11/2017 18:38:29	07/11/2017 18:36:32		8400.0	15646.0	8500	125	2017-11-07 18:38:29
8	1	07/11/2017 18:38:36	07/11/2017 18:36:32		6589.0	897.0	5000	125	2017-11-07 18:38:36
9	1	07/11/2017 18:38:46	07/11/2017 18:36:32	JPQ	1564.0	654.0	5000	125	2017-11-07 18:38:46
10	1	07/11/2017 18:38:56	07/11/2017 18:36:32	JPQ	7894.0	546.0	8500	125	2017-11-07 18:38:56
11	1	07/11/2017 18:39:06	07/11/2017 18:36:32	JPQ	1654.0	56465.0	9500	125	2017-11-07 18:39:06
12	1	07/11/2017 18:39:16	07/11/2017 18:36:32	JPQ	14250.0	45123.0	11000	125	2017-11-07 18:39:16
13	1	07/11/2017 18:39:26	07/11/2017 18:36:32	JPQ	1542.0	13133.0	12500	125	2017-11-07 18:39:26
14	1	07/11/2017 18:39:36	07/11/2017 18:36:32	JPQ	1231.0	13213.0	12800	125	2017-11-07 18:39:36
15	1	07/11/2017 18:39:46	07/11/2017 18:36:32	JPQ	1231.0	12546.0	13000	125	2017-11-07 18:39:46
16	1	07/11/2017 18:39:56	07/11/2017 18:36:32	JPQ	1985.0	1323.0	12800	125	2017-11-07 18:39:56
17	1	07/11/2017 18:40:06	07/11/2017 18:36:32	JPQ	9500.0	321.0	12500	125	2017-11-07 18:40:06
18	1	07/11/2017 18:40:16	07/11/2017 18:36:32	JPQ	9450.0	13213.0	13000	125	2017-11-07 18:40:16
19	1	07/11/2017 18:40:26	07/11/2017 18:36:32	JPQ	9468.0	3213.0	8000	125	2017-11-07 18:40:26
20	1	07/11/2017 18:40:36	07/11/2017 18:36:32	JPQ	9486.0	32132.0	9500	125	2017-11-07 18:40:36
21	1	07/11/2017 18:40:46	07/11/2017 18:36:32	JPQ	4897.0	12313.0	200	125	2017-11-07 18:40:46
22	1	07/11/2017 18:40:56	07/11/2017 18:36:32	JPQ	9462.0	31356.0	9500	125	2017-11-07 18:40:56
23	1	07/11/2017 18:42:08	07/11/2017 18:36:32	JPQ	9482.0	134548.0	11500	189	2017-11-07 18:42:08
24	1	07/11/2017 18:43:08	07/11/2017 18:36:32	JPQ	9463.0	5161.0	12530	189	2017-11-07 18:43:08

Tabla 1. Tabla obtenida con la importación a R desde la instrucción SQL.

Finalmente, los datos se deben filtrar debido a que, dependiendo del diseño de la tabla, existen muchos más registros con información adicional. Luego de filtrado se convierte los datos en serie de tiempo con semanas como unidad. La instrucción de filtrado se muestra a continuación.

```
x1 <- res[1:46,7]
x1
data1 <- ts(x1, start=c(2017,1), end=c(2017,length(x1)), frequency = 52)
```

Ilustración 13. Código para convertir los datos en serie de tiempo.

La primera línea de la figura 13, filtra seleccionando los elementos que están ubicados en los registros del 1 al 46 y que adicionalmente se encuentran en la columna 7, la cual está identificada como *temperatura_Retencion*.

La línea con la función ts(), recoge la información filtrada y la convierte en una serie de tiempo, que empieza con la primera semana del año 2017 (start = c(2017,1)) y finaliza la serie con la instrucción que cuenta la longitud de la serie (end = c(2017, length(x1)) y con la instrucción frecuency= 52, se determina la cantidad de semana en un año.

3.2 Código en R Para El Análisis de La Serie de Tiempo

Con el fin de realizar el objetivo del trabajo, se escribió un programa en R, basado en el diagrama de flujo definido en el inicio de la sección, este contempla varios pasos, los cuales se muestran a continuación:

- Análisis de diferenciaciones necesarias: en este paso, se usan dos funciones de R, que son ndiffs, y nsdiff; la primera devuelve el número de diferenciaciones que requiere la serie de tiempo, para eliminar la posible tendencia y volver la serie estacionaria, la segunda, sugiere el número de diferenciación para la eliminación de la estacionalidad que pueda contener la serie.

```
##=============================================================================
## Analisis de las diferenciaciones da al menos uno en los dos tipos
##=============================================================================

diferencias_d  <- ndiffs(data1)
Diferencias_D  <- nsdiffs(data1)
diferencias_d
Diferencias_D
```

Ilustración 14. Código para la sugerencia de diferenciaciones de la serie.

- Ciclo de diferenciaciones sugeridas: Para este punto, basados en los resultados de las funciones ndiffs y nsdiff, se crea una nueva serie de tiempo basada en la original, luego por medio de un ciclo FOR, se almacena la diferenciación de la serie en la variable auxiliar, y esto se repite el número de veces que se determina con las funciones. La variable h, contiene las diferenciaciones realizadas.

```
##segun sea el caso, el valor de las funciones ndiffs o nsdiffs, se usan para diferenciar
## los datos originales

h<-0
dff_curva <- data1
if(diferencias_d+Diferencias_D>0 ){
  for(h in 1:diferencias_d+Diferencias_D){
    dff_curva <- diff(dff_curva)
  }}
h
```

Ilustración 15. Generación de una serie de tiempo auxiliar con las diferenciaciones sugeridas.

-El paso siguiente, evalúa la curva según el test de Dickey Fuller, probando la Estacionaridad de la serie, devolviendo el resultado de la prueba y comparándola con el valor de test estadístico, dando el resultado como valido cuando el test, está por fuera de la región de aprobación de la hipótesis nula, si sucede lo contrario, el código vuelve a

diferenciar la serie, hace una nueva evaluación y si en este segundo caso ya se cumple el test estadístico, la prueba se pasa y se toman los datos de las diferenciaciones realizadas.

```
modelo_valido<-"No_OK"
while(modelo_valido != "OK"){

  #test dicken -fuller, para determinar si la serie es estacionaria, si si, se puede aplicar metodo box-jenkins.
  plot(ur.df(dff_curva, type = "none", lags = 12 ))
  resultado_curva<-summary(ur.df(dff_curva, type = "none", lags = 1 ))
  resultado_curva
  resultado_curva@cval
  resultado_curva@teststat[1]
  resultado_curva@cval[,2]


  if(resultado_curva@teststat[1]< resultado_curva@cval[,2]){
    modelo_valido <- "OK"
    par(mfrow=c(2,3))
    plot(data1,type="o")
    acf(data1, lag.max = 50)
    pacf(data1, lag.max = 50)
    plot(dff_curva ,type="o")
    abline(h=2*sqrt(var(dff_curva)),col="red",lty=2)
    abline(h=-2*sqrt(var(dff_curva)),col="red",lty=2)
    acf(dff_curva, lag.max = 50)
    pacf(dff_curva, lag.max = 50)
  }else{
    modelo_valido <-"NO_OK"
    dff_curva <- diff(dff_curva)
    h = h+1
    diferencias_d<-h
  }
}
modelo_valido
h
```

Ilustración 16. Bucle para evaluación de la serie y determinar automáticamente la Estacionaridad.

- Con la verificación automática de Estacionaridad, se procede con el con el código para la iteración de parámetro p y q del modelo ARIMA, con el cual se buscan los valores que permitan pasar los test estadísticos de Jarque Bera (normalidad) y el Ljung Box(de independencia) de tal manera que definiendo los valores mínimos de estos dos test, se da por aprobada la serie para realizar el respectivo pronóstico.

```
parametro_p<-1
parametro_q<-1
if (Diferencias_D > 0 ){
  parametro_P<-1
  parametro_Q<-0
}
modelo_aprobado <- "NO_OK"
while(modelo_aprobado != "OK"){
  if (Diferencias_D > 0 ){
    modelo1<-stats::arima(data1,order=c(parametro_p,diferencias_d,parametro_q),
                          seasonal = c(parametro_P,Diferencias_D,parametro_Q),
                          method = c("CSS-ML", "ML", "CSS"))
  }else{
    modelo1<-stats::arima(data1,order=c(parametro_p,diferencias_d,parametro_q),method = c("CSS-ML", "ML", "CSS"))
  }
  modelo1$var.coef
  modelo1$coef
  modelo1$residuals
  modelo1

  BIC_r<-BICt(modelo1$loglik,(length(modelo1$coef)+1),modelo1$nobs)
  et<-residuals(modelo1)
  data1.fit<-data1-et

  par(mfrow=c(3,2))
  plot(data1,type="l",lty=2)
  lines(data1.fit,type="l",col="red")
  plot(scale(et),type="l",main="Residuales")
  abline(h=2*sqrt(var(scale(et))),col="red",lty=2)
  abline(h=-2*sqrt(var(scale(et))),col="red",lty=2)
  acf(et)
  pacf(et)
  qqPlot(scale(et))
  acf(et^2)
  #tsdiag(modelo1)
  resultado_box<-Box.test(et, lag=45, type="Ljung-Box")
  resultado_box$p.value
  resultado_jaque<-jarque.bera.test(et)
  resultado_jaque$p.value
  resultado_jaque

  if ((resultado_box$p.value > 0.50) && (resultado_jaque$p.value > 0.05)){
  modelo_aprobado <- "OK"
  }else{
    parametro_p <- parametro_p +1
    parametro_q <- parametro_q +1
    modelo_aprobado <- "NO_OK"
  }
}
modelo_aprobado
```

Ilustración 17. Bucle iterativo, para encontrar los valores p, q del modelo ARIMA que satisfagan los test estadísticos.

3.4 Generación y Exportación de Resultados

Debido a que el código evalúa automáticamente los resultados del modelo de manera estadística por medio de los test de normalidad y de independencia, se verifica que estos se cumplan con la variable "*modelo aprobado"*. Si esta variable contiene la cadena "OK", significa que existe evidencia estadísticamente aceptable para que el modelo generado pueda describir un pronóstico. Por lo tanto, el paso siguiente se encarga de generar el pronóstico con una confianza del 95%. Estos resultados quedan consignados en la variable *pred4*.

```
modelo_aprobado
modelo1$coef
modelo1
resultado_box
resultado_jaque
BIC_r
##============================================================================
## Pronostico del modelo
##============================================================================
pred4<- forecast(modelo1, h=12, level = 95)
par(mfrow=c(1,1))
plot(pred4)

abline(h=mean(data1),
       col="red")
abline(h=(mean(data1)-2*sd(data1)),
       col="blue")
abline(h=(mean(data1)+2*sd(data1)),
       col="blue")
abline(h=9500,
       col="orange")
```

Ilustración 18. Generación de pronóstico para el modelo seleccionado.

Esta variable con el pronóstico, ya validado estadísticamente, se exporta a la base de datos por medio de la instrucción sqlQuery(), para que esto se pueda hacer, debe existir una tabla con las columnas definidas por el pronóstico, estas son: *Pronostico_low*, *pronostico_mean* y *pronostico_up*. Esta tabla se borra previamente para

que no se mezclen con pronósticos anteriores; ya que, la idea es actualizar los pronósticos en la medida que la serie de tiempo cambia.

```
##============================================================================
## ESTAPA FINAL DEL PROGRAMA, DONDE LOS RESULTADOS SON ESCRITOS EN LA BASE DE
## DATOS, PREVIO A ESTO SE BORRAN LOS DATOS DE PRONOSTICO ANTERIORES
##============================================================================

#BOrrar resultados previos
sqlQuery(dbhandle, "DELETE FROM Resultados_R")

#ingresar los datos del resultado en la tabla de resultados de la base de datos SQL

for(h in 1:length(pred4$mean)){
  msg_txt<- paste("INSERT INTO Resultados_R VALUES(",
                  pred4$lower[h],",", pred4$lower[h], "," ,  pred4$lower[h], ")" )
  sqlQuery(dbhandle, msg_txt)
}
odbcClose(dbhandle)
```

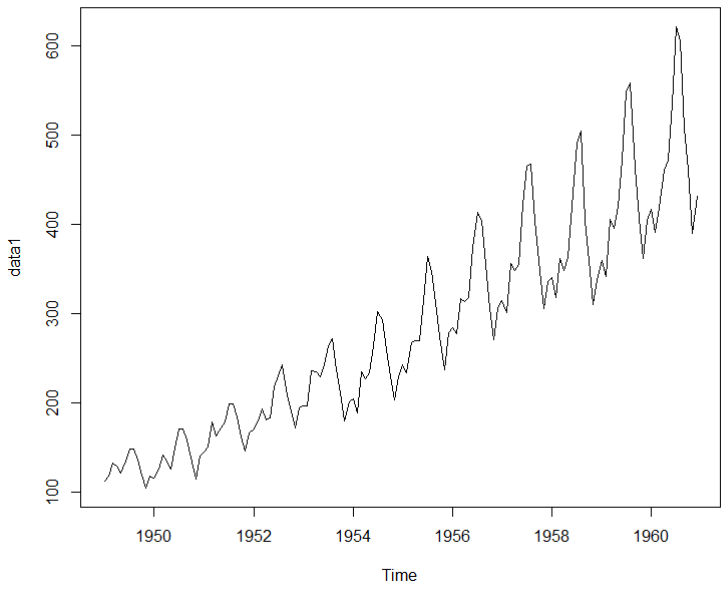
Ilustración 19. Código para la exportación de la información a la tabla de la base de datos.

Capítulo 4. Análisis y Resultados

Una vez el procedimiento esta creado, se procede con la evaluación de un modelo conocido, el cual se basa en la serie de tiempo que tiene R *AirPassangers*, luego de esta validación se prueba el código con la serie referida en el inicio del capítulo 3.

4.1 Evaluación del Programa Con Datos Conocidos.

En la evaluación del modelo conocido *AirPassangers*, no se tiene en cuenta la conectividad con el motor de base de datos SQL, por lo que se inicia con los resultados obtenidos por la ejecución del código mismo, partiendo de la importación de dicha serie.




```
data("AirPassengers")
#data("airmiles")
x1<- AirPassengers
#x1<- airmiles
data1 <- x1
plot(data1)
```

Ilustración 20. Serie de tiempo conocida para evaluar el desempeño del programa.

Al ejecutar el primer código obtenemos el número de diferenciaciones sugeridas por los comandos ndiffs y nsdiffs.

```
> h<-0
> dff_curva <- data1
> if(diferencias_d+Diferencias_D>0 ){
+   for(h in 1:diferencias_d+Diferencias_D){
+     dff_curva <- diff(dff_curva)
+   }}
> h
[1] 2
>
```

```
> diferencias_d  <- ndiffs(data1)
> Diferencias_D  <- nsdiffs(data1)
> diferencias_d
[1] 1
> Diferencias_D
[1] 1
>
```

```
> resultado_curva@cval
      1pct  5pct 10pct
tau1 -2.58 -1.95 -1.62
> resultado_curva@teststat[1]
[1] -8.677137
> resultado_curva@cval[,2]
[1] -1.95
> h
[1] 2
```

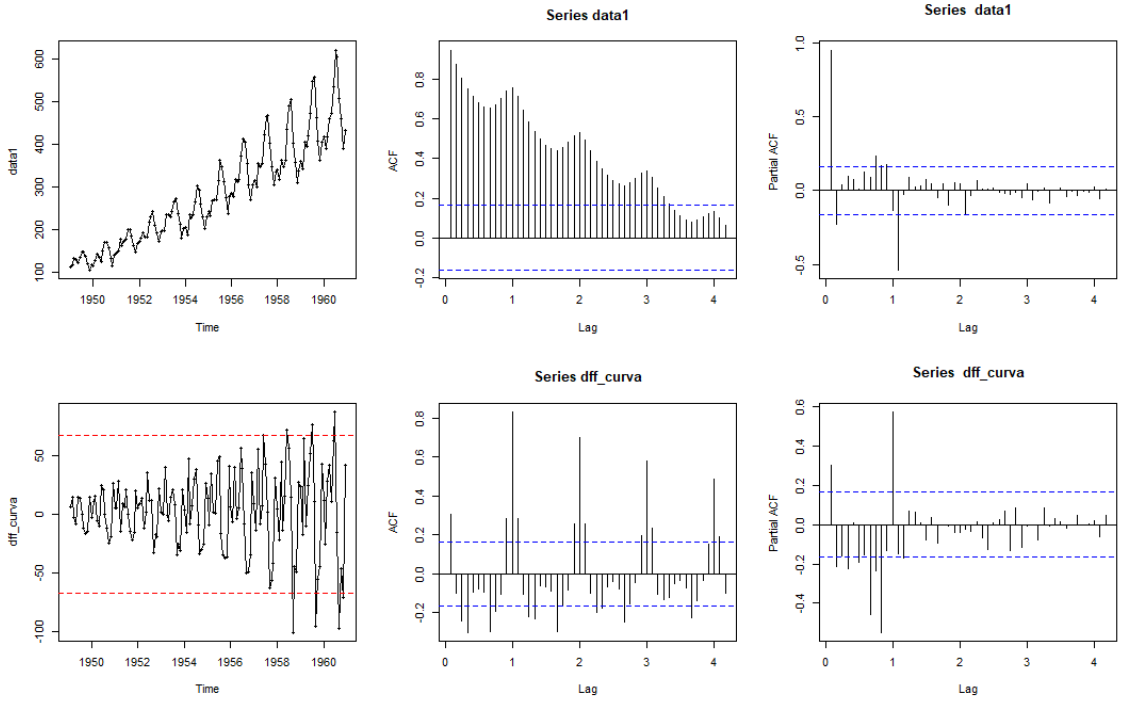


Ilustración 21.Resultados del código ejecutado con el programa planteado.

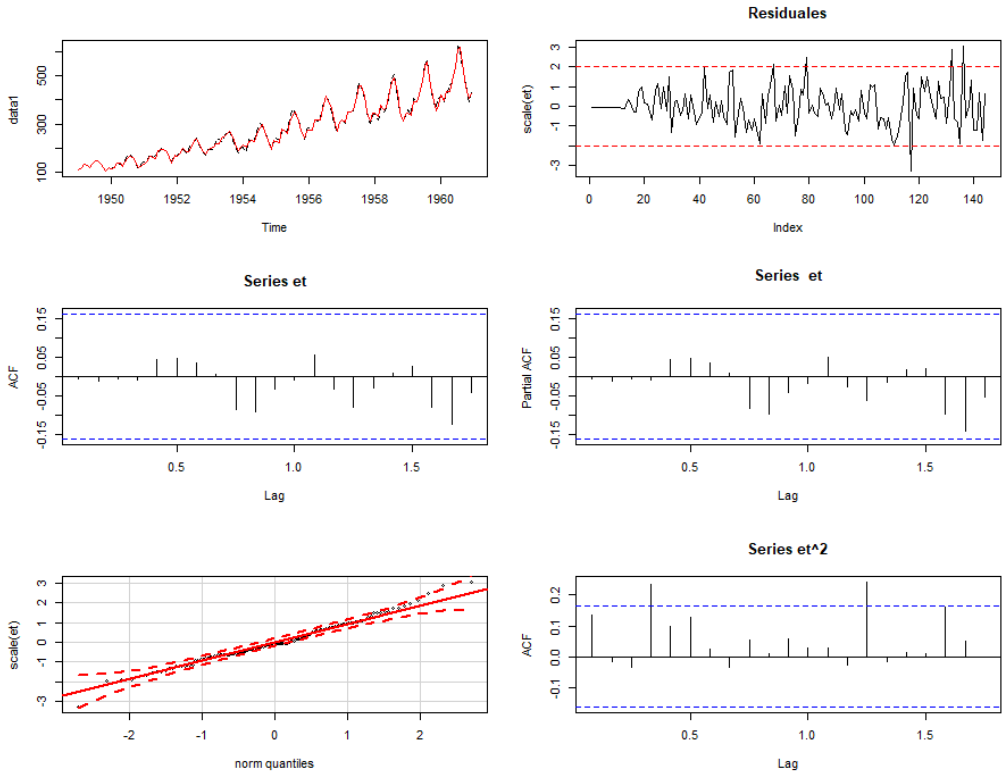


Ilustración 22. Diagnóstico de correlaciones y normalidad resultado de la iteración en el programa.


```
> resultado_box

        Box-Ljung test

data:  et
X-squared = 32.611, df = 45, p-value = 0.9158

> resultado_jaque

        Jarque Bera Test

data:  et
X-squared = 4.8012, df = 2, p-value = 0.09066

> BIC_r
[1] 1061.747
```

Ilustración 23. Resultados de los test estadísticos generados.

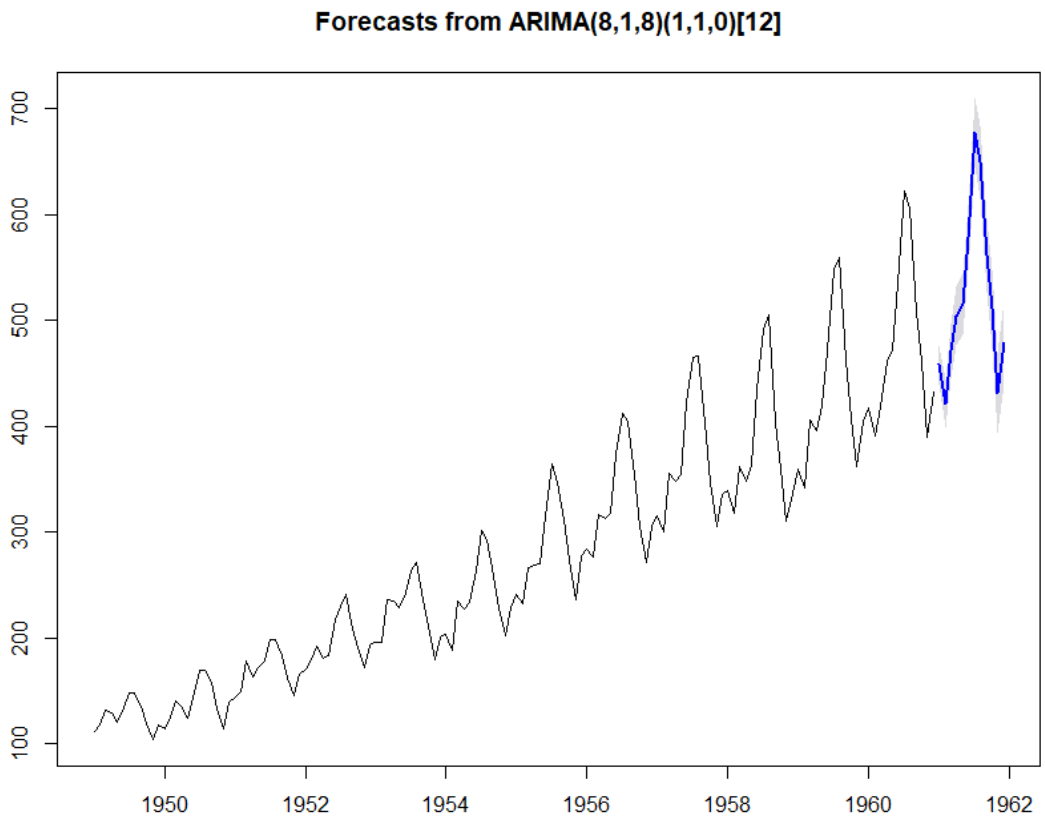


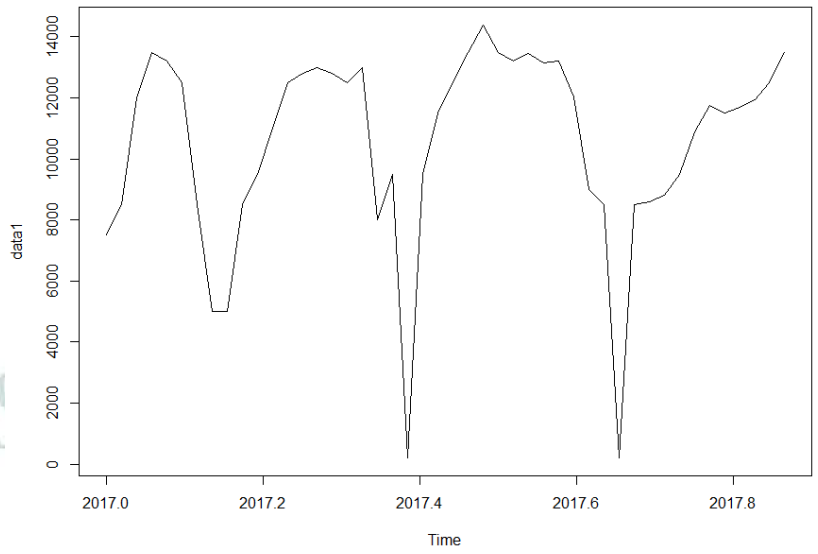
Ilustración 24. Pronóstico generado por el programa con un modelo ARIMA(8,1,8)(1,1,0)[12]

Los gráficos anteriores muestran y validan el procedimiento, encontrando un modelo ARIMA(8,1,8)(1,1,0)[12], por medio de la iteración del programa, el cual incrementa los valores de p y q simultáneamente para optimizar el código y la velocidad de respuesta. También podemos observar que, para este modelo, los test estadísticos validan la confiabilidad del modelo, con un p-value del test Ljung Box del 91% (el modelo no es

aleatorio) y con un p-value en el test Jarque Bera del 9%, indicando que el modelo pasa el test de normalidad.

4.2 Evaluación del Programa Con La Serie de Tiempo Propuesta.

Ya con el código validado con una serie conocida, se procede a realizar la ejecución con la serie propuesta.




```
##=============================================================================
## CONEXION CON BASE DE DATOS SQL
## la conexion a la base de datos, se debe realizar primero por la configuracion de un DSN,
## esta configuracion se hace por medio
## de las herramientas de admisnitracion de windows
## tambien se crea la serie de tiempo, usando semanas como base.
##=============================================================================

dbhandle <- odbcConnect("Origen_Datos")
res <- sqlQuery(dbhandle, 'SELECT TOP(100) * FROM temperaturas')
res


x1 <- res[1:46,7]
x1
data1 <- ts(x1, start=c(2017,1), end=c(2017,length(x1)), frequency = 52)
plot(data1)
```

Ilustración 25. Serie recolectada de la base de datos, convertida a serie de tiempo y graficada.

```
> diferencias_d  <- ndiffs(data1)
> Diferencias_D  <- nsdiffs(data1)
> diferencias_d
[1] 0
> Diferencias_D
[1] 0
>
```

Ilustración 26. Las diferencias sugeridas según las funciones ndiffs y nsdiffs.

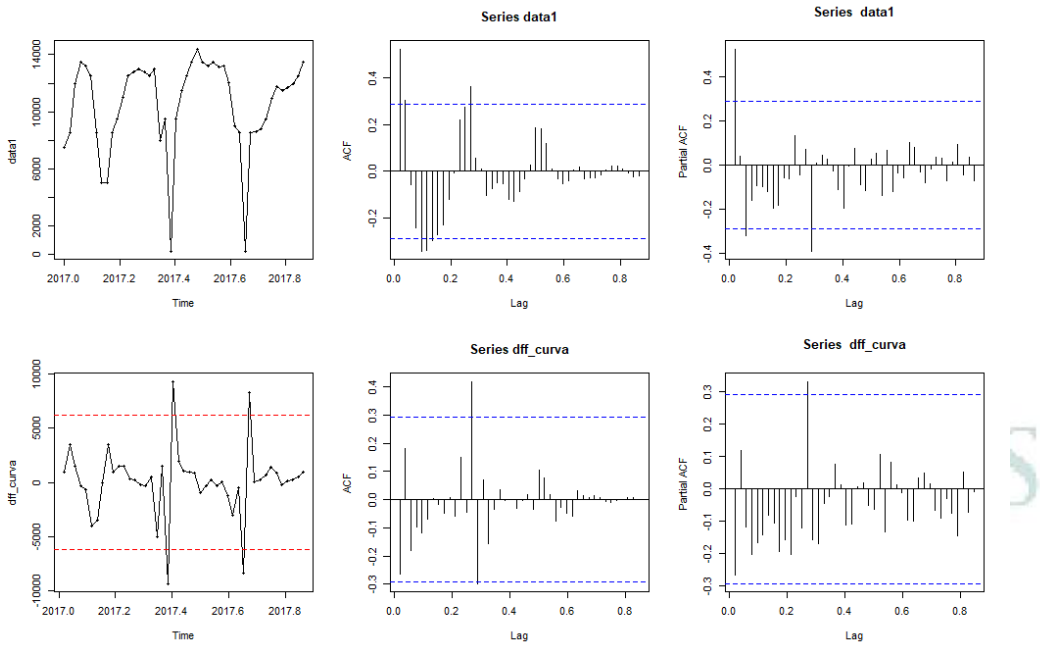


Ilustración 27. El diagnóstico, después de ejecutar el programa y realizar el bucle para determinar las diferenciaciones.

Como se observa en la ilustración 27, el código valida las series y determina que para pasar el test estadístico Dickey-Fuller, se requiere realizar una diferenciación.

```
> modelo_valido
[1] "OK"
> resultado_curva@cval
      1pct  5pct 10pct
tau1 -2.62 -1.95 -1.61
> resultado_curva@teststat[1]
[1] -4.634531
> resultado_curva@cval[,2]
[1] -1.95
> h
[1] 1
>
```

Ilustración 28. Resultados del test estadístico Dickey-Fuller.

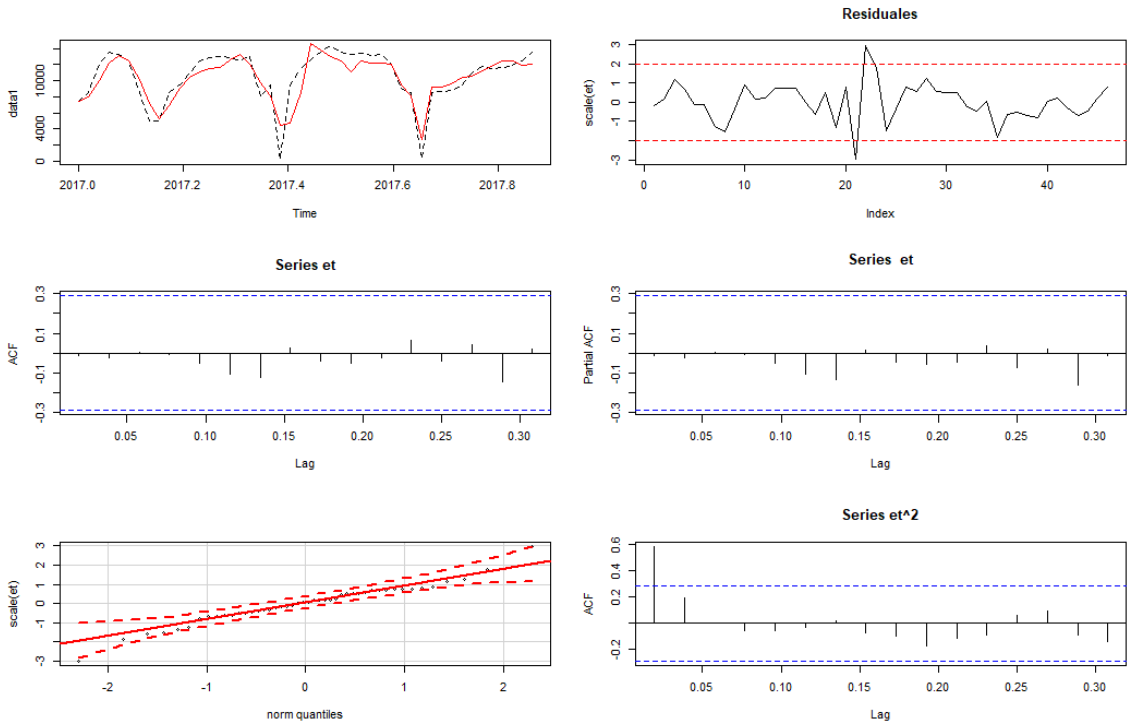


Ilustración 29. Evaluación del modelo generado.

El resultado de la iteración del programa arroja un modelo ARIMA(13,1,13), el cual pasa los test de normalidad e independencia, y el programa lo clasifica como valido para realizar el pronóstico.

```
Call:
stats::arima(x = data1, order = c(parametro_p, diferencias_d, parametro_q),
    method = c("CSS-ML", "ML", "CSS"))

Coefficients:
          ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8      ar9     ar10     ar11     ar12     ar13     ma1      ma2
      -1.1872  -1.4272  -1.5648  -1.589  -1.5639  -1.4707  -1.3159  -1.1683  -0.9694  -0.8253  -1.031  -0.7626  -0.6099  0.681  0.9660
s.e.      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN   0.2039      NaN      NaN   0.1386      NaN  0.3533
         ma3     ma4     ma5      ma6      ma7      ma8      ma9     ma10     ma11     ma12     ma13
      0.7460  0.4699  0.0363  -0.3326  -0.6366  -0.8888  -0.9756  -0.8009  0.0367  -0.3544  0.0941
s.e.  0.3986  0.4307  0.3839   0.5029   0.3123   0.3421   0.3987      NaN  0.3924      NaN      NaN

sigma^2 estimated as 2348573:  log likelihood = -406.85,  aic = 867.7
```

Ilustración 30. Coeficientes generados por el programa.

```
> resultado_box

        Box-Ljung test

data:  et
X-squared = 14.153, df = 45, p-value = 1

> resultado_jaque

        Jarque Bera Test

data:  et
X-squared = 5.3106, df = 2, p-value = 0.07028

>
```

Ilustración 31. Valores de los test de normalidad e independencia.

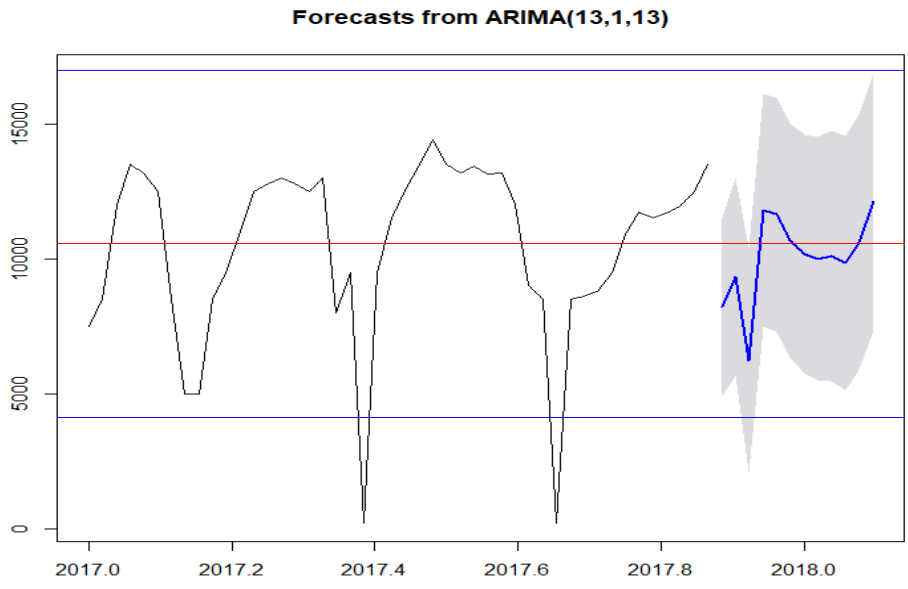


Ilustración 32. Pronóstico generado.

4.3 Comparación de La Serie Original Con El Valor de Pronóstico

En la figura 1 del capítulo 3, se puede observar la serie original y completa, la cual fue fraccionada, para comparar el pronóstico encontrando y determinar cómo se comporta el programa propuesto. La línea negra muestra el punto donde se fracciona la serie.

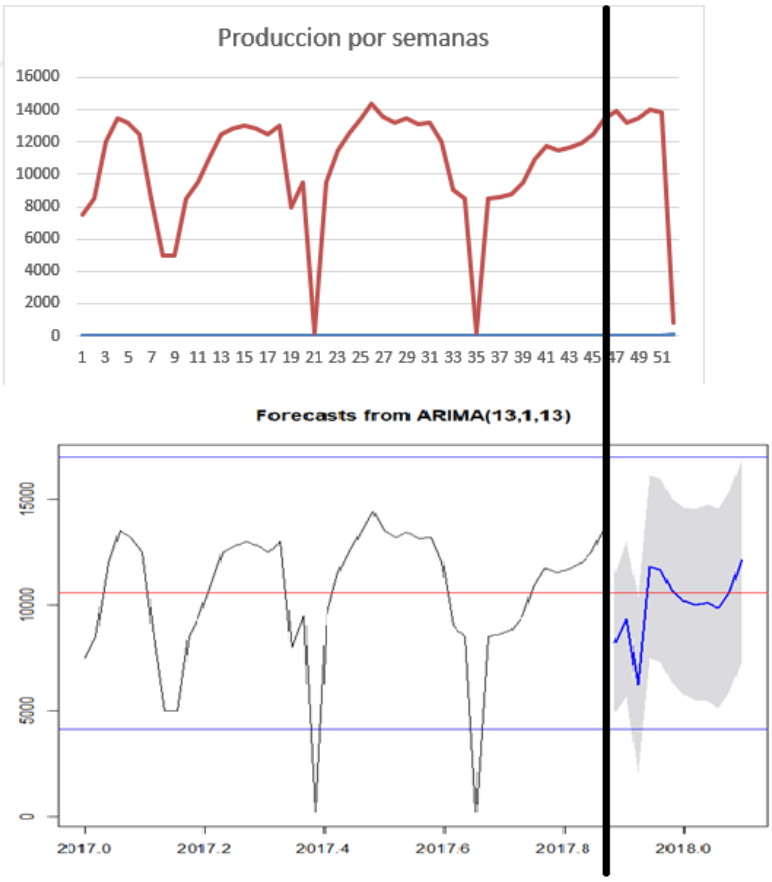


Ilustración 33. Comparación de la serie original con el pronóstico.

El pronóstico generado, detecta en una medida significativa la tendencia de la curva original; aunque, no en las mismas magnitudes. Como ejercicios adicionales, se modificaron los parámetros de inicio de p y q, con valores más altos, buscando que se tuviera información de más rezagos en el modelo y verificando el pronóstico con la serie original. Encontrando un modelo ARIMA(31,1,21) con la siguiente evaluación:

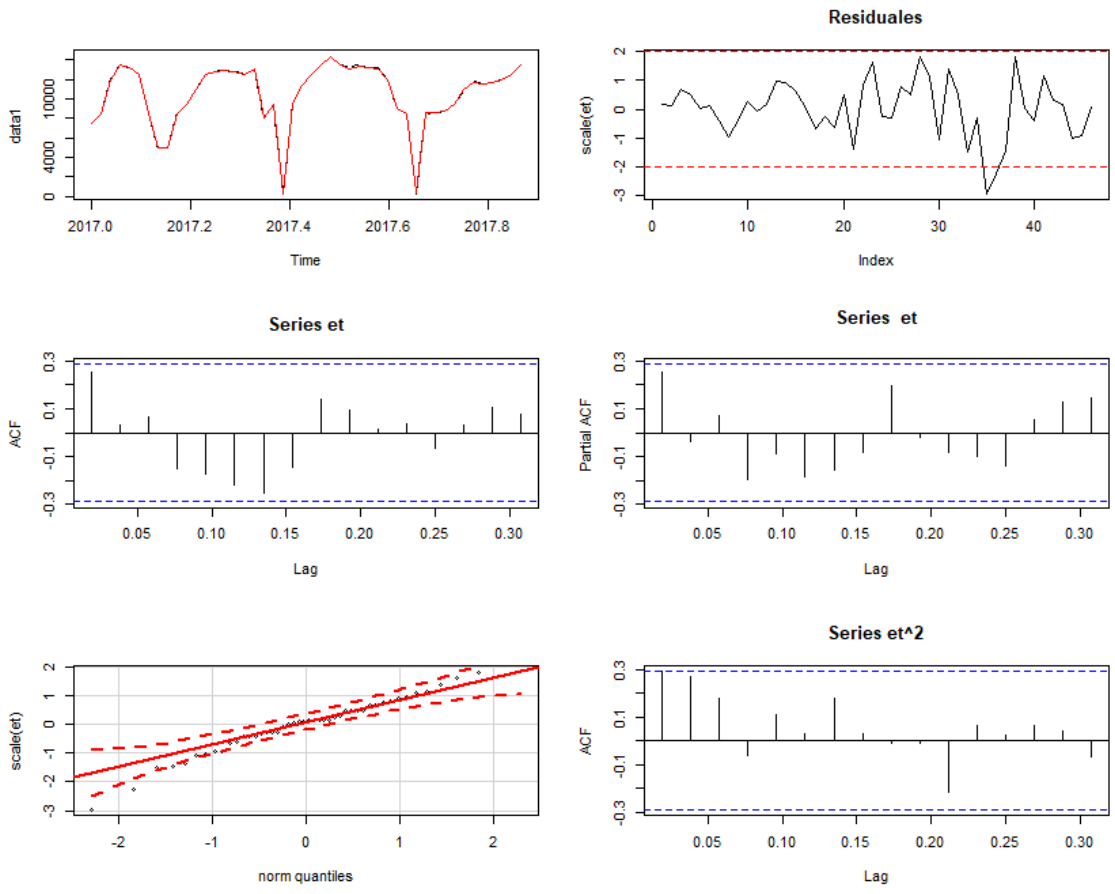


Ilustración 34. Evaluación del modelo encontrado, y sus gráficos de correlación.

```
> resultado_box

        Box-Ljung test

data:  et
X-squared = 39.334, df = 45, p-value = 0.71

> resultado_jaque

        Jarque Bera Test

data:  et
X-squared = 3.2748, df = 2, p-value = 0.1945
```

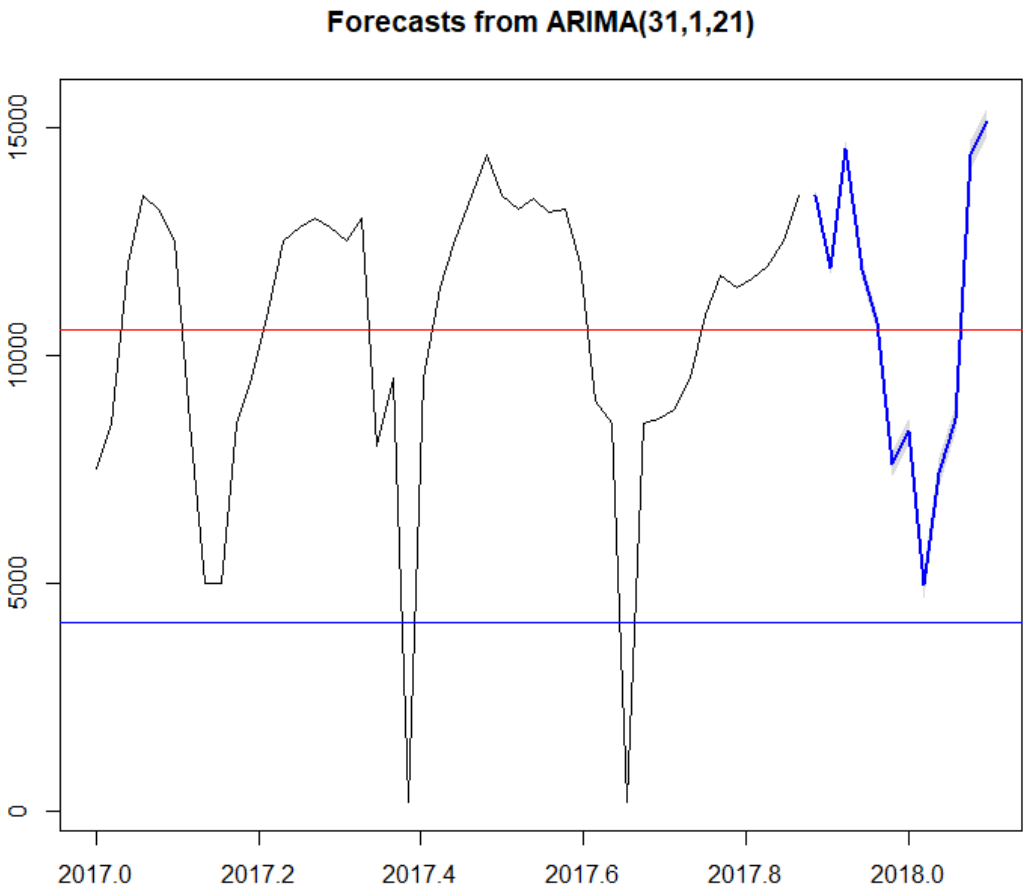


Ilustración 35. Pronóstico para modelo ARIMA(31,1,21)

Con valores más altos de los valores p y q, se logra incorporar más información al modelo, logrando una banda de confianza más estrecha y al mismo tiempo una mejor aproximación a la serie original.

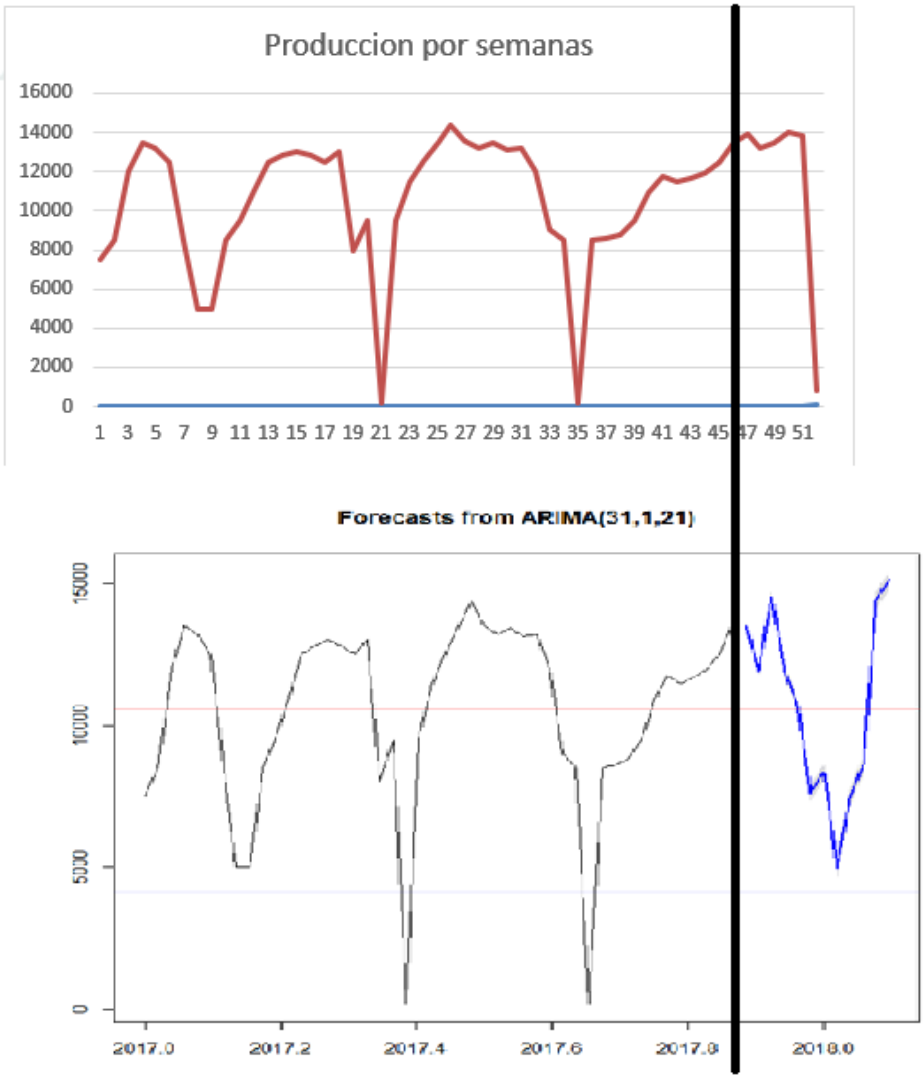


Ilustración 36. Comparación de la serie original y el pronóstico para modelo ARIMA (31,1,21)

Capítulo 6. Conclusiones

De acuerdo con los resultados expuestos en el capítulo 4, podemos realizar las siguientes conclusiones sobre el proyecto presentado:

- La conectividad de R con fuentes de bases de datos es robusta, con librerías como RODBC, y con la funcionalidad de Windows, permitiendo extender las funcionalidades de R como herramienta estadística, a otras aplicaciones.
- R ofrece herramientas estadísticas y de programación óptimas para la realización de evaluación de modelos interactivos, con los que podemos evaluar en tiempo real series de tiempo con base en pocos parámetros encontrando pronósticos estadísticamente correctos.
- Los modelos ARIMA, son una herramienta poderosa al momento de la generación de pronósticos de una manera confiable; ya que, con una sola variable podemos evaluar información que de alguna manera es externa a esta variable al encontrarse incorporada indirectamente, haciendo robusta la metodología.
- Con el lenguaje R, se pueden diseñar herramientas para el análisis estadístico, que al automatizarlas reducen tiempo y aumentan la velocidad con que se pueden tener resultados para la toma de decisiones.

- En el programa generado, se evidencia una afectación del resultado, al cambiar los valores p y q; ya que, en el programa se incrementan estos valores simultáneamente, aunque los valores iniciales pueden ser diferentes y de esta manera generar nuevos modelos válidos, o en el peor de los casos, generando errores en la ejecución del código.

- La validación del código generado, complemento el trabajo, en el sentido que se pudo comparar el resultado del programa, con una serie conocida y comparar los resultados de cada uno de ellos.

- El algoritmo diseñado, tuvo un buen comportamiento al poder determinar modelos ARIMA para diferentes series de tiempo, lo que indica que se puede seguir mejorando para poder trabajar automáticamente con una gran variedad de series de tiempo.

Para la continuación y mejora del presente trabajo se hacen las siguientes recomendaciones:

- Profundizar en la selección de los valores p,q en el modelo ARIMA para utilizar la función de una manera más eficiente.

- Realizar varias pruebas con diferentes series de tiempo para validar el procedimiento bajo diferentes condiciones, así poder detectar las falencias y corregirlas.

- Explorar otros motores de bases de datos, con otros tipos de información, e incorporar diferentes modelos.

- En R existen otros métodos que, para el caso de este trabajo, se pudieron usar con resultados poco confiables, como el Fuzzy Logic y Redes Neuronales, por lo que se recomienda profundizar en el uso de estas funciones en la predicción en tiempo real.

Capítulo 7. Referencias

1 *CHRIS CHATFIELD*: *The Analysis of Time Series. An Introduction*. 5th edition, VII + 283 pp. Chapman & Hall 1996

2 DURAN, LUIS: Bases de datos con Visual Basic. Primera Edicion, 289 p. MARCOMBO EDICIONES TECNICAS, ALFA OMEGA EDITOR SA, 2007

3 https://cran.r-project.org/web/packages/RODBC/RODBC.pdf

Author

Brian Ripley [aut, cre],

Michael Lapsley [aut] (1999 to Oct 2002)

Anexo 1. Código en R

```
##=============================================================================
===

##PROGRAMA DE ANALISIS DE DATOS, TOMADOS DE UN MOTOR DE BASE DE DATOS TIPO SQL
EXPRESS

##=============================================================================
===



library(urca)

library(forecast)

library(car)

library(TSA)

library(FitAR)

library(RODBC)

install.packages("taskscheduleR")



##FUNCION BICt para el calculo del BICT mas bajo para la aplicacion de un modelo ARMA



BICt <- function(L,k,n){
```

```
 -2*L + k*log(n)

}


#CONEXION CON BASE DE DATOS SQL

#la conexion a la base de datos, se debe realizar primero por la configuracion de un DSN, esta configuracion se hace por medio

#de las herramientas de admisnitracion de windows


dbhandle <- odbcConnect("database_")

res <- sqlQuery(dbhandle, 'SELECT TOP(100) * FROM temperaturas')

res


x1<- AirPassengers


data1 <- x1


#columan 5 contiene los datos que se van a analiza

x1 <-res[1:46,7]
```

```
plot(res[,7], type="l")

length(x1)

x1

data1 <- ts(x1, start=c(2017,1), end=c(2017,length(x1)), frequency = 52)

plot(data1)



Diferencias_D=0

diferencias_d  <- ndiffs(data1)

Diferencias_D  <- nsdiffs(data1)

diferencias_d

Diferencias_D

#identificacion del modelo segun los datos

plot(data1,type="o")

abline(h=mean(data1),

     col="red")

abline(h=(mean(data1)-2*sd(data1)),

     col="blue")

abline(h=(mean(data1)+2*sd(data1)),

     col="blue")

abline(h=9500,

     col="Orange")
```

```
#crear la curva diferenciada segun el comando ndffs

dff_curva <- data1



if(diferencias_d+Diferencias_D>0 ){

for(h in 1:diferencias_d+Diferencias_D){

 dff_curva <- diff(dff_curva)

 }}

h

plot(dff_curva)



#identificacion del modelo

par(mfrow=c(2,3))

plot(data1,type="o")

acf(data1, lag.max = 50)

pacf(data1, lag.max = 50)

plot(dff_curva ,type="o")

abline(h=2*sqrt(var(dff_curva)),col="red",lty=2)

abline(h=-2*sqrt(var(dff_curva)),col="red",lty=2)

acf(dff_curva, lag.max = 50)

pacf(dff_curva, lag.max = 50)
```

```
#test adf

plot(ur.df(dff_curva, type = "none", lags = 12 ))

resultado_curva<-summary(ur.df(dff_curva, type = "none", lags = 1 ))

resultado_curva

resultado_curva@cval

resultado_curva@teststat



#El resultado de la funcion summary, ur.df (prueba dicken-fuller)



#evaluacion del modelo

modelo1 <- auto.arima(data1, d=diferencias_d, max.p = 30,max.q=30, D= Diferencias_D, max.P = 30, max.Q = 30 )

modelo1$var.coef

modelo1$coef

modelo1$residuals

modelo1



##Parte adicional de funciones vistas en clase

### Modelamiento por Fuerza Bruta



Diferencias_D=0
```

```
diferencias_d  <- ndiffs(data1)

Diferencias_D  <- nsdiffs(data1)

diferencias_d

Diferencias_D



p.max <- 30

q.max <- 30

k <- 1

BIC <- matrix(0,((p.max+1)*(q.max+1)),3)

for(p in 15:(p.max+1)){

 for(q in 15:(q.max+1)){

  modelo<-try(stats::arima(data1,order=c((p-1),diferencias_d,(q-1))),silent=TRUE)

  if('try-error' %in% class(modelo)){ next

  }else{

   logL<-modelo$loglik

   BIC[k,1]<- -2*logL + (length(modelo$coef)+1)*log(modelo$nobs)

   BIC[k,2]<- p-1

   BIC[k,3]<- q-1

   resultado_modelo <- summary(ur.df(modelo1, type = "none", lags = 1 ))

   BIC[k,4] <- resultado_modelo

  }

  k <- k+1
```

```
  }

}



resultado_modelo

mean(BIC[,1])

BIC



ref= 100000

pos=0

for(h in 2:length(BIC[,1])){

 if ((BIC[h,1] < ref)&&(BIC[h,1] != 0)){

   ref= BIC[h,1]

   pos=h

 }

}

ref

BIC[pos,]

BIC[pos, 2]

BIC[pos, 3]



ts.plot(modelo1)
```

```
modelo1 <- stats::arima(data1,order=c(30,1,30))

BICt(modelo1$loglik,(length(modelo1$coef)+1),modelo1$nobs)

ts.plot(modelo1)




##Pronostico de la funcion

pred4<- forecast(modelo1, h=12, level = 95)


plot(pred4)


abline(h=mean(data1),
       col="red")
abline(h=(mean(data1)-2*sd(data1)),
       col="blue")
abline(h=(mean(data1)+2*sd(data1)),
       col="blue")
abline(h=9500,
       col="Orange")



pred4<- forecast(modelo1, h=12, level = 90)
```

```
plot(pred4)

pred4$lower

pred4$mean

pred4$upper



#BOrrar resultados previos

sqlQuery(dbhandle, "DELETE FROM Resultados_R")



#ingresar los datos del resultado en la tabla de resultados de la base de datos SQL



for(h in 1:length(pred4$mean)){

  msg_txt<- paste("INSERT INTO Resultados_R VALUES(",

          pred4$lower[h],",", pred4$lower[h], "," ,  pred4$lower[h], ")" )

  sqlQuery(dbhandle, msg_txt)

}

odbcClose(dbhandle)
```