UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ADE
FACULTAD DE ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS. UPV

Utilización de la base de datos Eurostat. Elaboración de informes automáticos con R

Trabajo de fin de grado

Grado en Administración y Dirección de Empresas

Autor: Fernando Reboyras Soleto

Tutores: Dr. Francisco Javier Ribal Sanchis
Dra. Ana María Debón Aucejo

Curso: 2019-2020

Índice general

Índice de figuras

Introducción

En este capítulo introductorio hablaremos de los motivos, objetivos y contexto de realización del presente trabajo.

Motivación

Lo que se busca es obtener conocimientos y práctica en el área del *Business Analytics*, que se puede definir como «el uso de datos para tomar decisiones de negocio más sólidas y basadas en evidencias» (Seddon, Constantinidis, Tamm, y Dod, 2017). Dicha disciplina está relacionada con asignaturas como Macroeconomía, Dirección Financiera, Análisis y Consolidación Contable o Economía Financiera, que se apoyan en el estudio de datos socioeconómicos para tomar decisiones de carácter empresarial.

Sin embargo, dicha información debe cumplir al menos dos criterios: proceder de fuentes fiables y ser presentada de forma que pueda ser interpretada.

En este trabajo estudiaremos una fuente que permite cumplir el primer criterio: la extensa base de datos europea Eurostat, ya que se trata de una organización oficial y supranacional que sigue rigurosos procedimientos para almacenar datos.

Por otra parte, para presentar la información en informes le aplicaremos un tratamiento previo mediante la ciencia de datos, que según Oracle es «un campo interdisciplinario que utiliza métodos, procesos, algoritmos y sistemas científicos para extraer valor de los datos» (Oracle, 2020). Para ello también deberemos estudiar R, que es un potente lenguaje y entorno de programación diseñado para computación estadística y gráfica (R Core Team, 2020). Esta otra disciplina se relaciona con asignaturas de carácter estadístico como Métodos Estadísticos en Economía o Econometría.

Dado que trataremos con datos recientes, nacionales y supranacionales, tendremos la oportunidad de observar la situación actual de los países en aspectos económicos, sociales e incluso midiendo el cumplimiento de los Objetivos de Desarrollo Sostenible (en adelante ODS) propuestos por la Organización de las Naciones Unidas (en adelante ONU).

Adicionalmente, mostraremos un ejemplo de utilidad añadida realizando previsiones sobre los conjuntos de datos por medio de modelos estadísticos.

Objetivos

- Aprender a gestionar bases de datos económico-sociales de gran fiabilidad en el ámbito internacional, como es Eurostat
- Extraer información de las bases aplicando gráficos descriptivos y modelos sencillos a través del lenguaje de programación R con el entorno RStudio
- Aplicar conocimientos de las asignaturas de estadística para realizar predicciones mediante modelos ARIMA
- Manipular datos económicos y sociales de actualidad relacionados con los conocimientos adquiridos durante el grado
- Automatizar la extracción de datos para trabajar con información histórica y contemporánea
- Desarrollar una aplicación interactiva e intuitiva para el usuario que presente los datos y facilite la interpretación

Contexto

Vivimos en una época de importante globalización y desarrollo tecnológico, en la que la información se encuentra al alcance de todos. Cualquiera con curiosidad puede encontrar respuestas a sus preguntas en páginas como Google o en foros online. Incluso cuando se trata de información de carácter estadístico, existen numerosas bases de datos con acceso online gratuito. Sin embargo, es altamente recomendable escoger únicamente fuentes fiables. Por ejemplo, a nivel nacional en España disponemos principalmente del Instituto Nacional de Estadística y a nivel europeo Eurostat es una de las más conocidas.

Por lo tanto, la problemática ya no gira en torno al acceso a los datos, sino a su tratamiento e interpretación, cuyas disciplinas respectivas son la ciencia de datos y el *Business Analytics*.

La primera es de suma importancia, ya que en las bases de datos la información se optimiza para su almacenamiento, no para su lectura e interpretación. Afortunadamente, existen numerosos recursos de apoyo en internet para facilitar el aprendizaje de la ciencia de datos. Un buen ejemplo es el libro de Wickham y Grolemund (2017), que introduce tanto al lenguaje de programación R como a la ciencia de datos y se encuentra completamente disponible de forma gratuita como página web.

La segunda es esencial, ya que el uso de la información en el ámbito empresarial no es el futuro, sino el presente. Esto se debe a que los datos pueden ser utilizados, por ejemplo, para «detectar nuevas oportunidades, identificando nichos de mercado, y desarrollando nuevos productos y servicios» (Acito y Khatri, 2014).

Por añaditura, la reciente pandemia causada por el COVID-19 ha sacudido prácticamente todos los sectores, incluidos el económico y social. Por lo tanto, el estudio de la evolución reciente de los países mediante indicadores tiene una importancia añadida.

Finalmente, como estudiante del Doble Grado de Ingeniería Informática y ADE, este trabajo también resulta de un interés suplementario. Como explican Acito y Khatri (2014), «extraer valor de los datos requiere alinear estrategia y comportamientos deseables con la gestión del rendimiento del negocio en conjunción con tareas y capacidades analíticas». En otras palabras, el *Business Analytics* integra el tratamiento de datos socioeconómicos con su interpretación, dando uso a conocimientos y habilidades adquiridos en ambas carreras.

1. Introducción a Eurostat

Este primer capítulo tiene como objetivo dar una presentación de la base de datos europea. Para ello, hablaremos de su historia desde su origen hasta hoy y de la estructura de los datos que presenta.

1.1 Origen y evolución de Eurostat

También conocida como la Oficina Europea de Estadística, Eurostat fue creada en 1953 para cumplir los requisitos de la Comunidad del Carbón y el Acero, actuando como su División de Estadística.

Desde entonces ha seguido recopilando datos de los países miembros de la Unión Europea, que también nombraremos como UE. De la misma forma que ésta se iba expandiendo y desarrollando, Eurostat ha ido ampliando sus áreas de investigación y adaptando sus políticas para alinearse con las directrices y objetivos de la UE.

En 1970, Eurostat estableció la Nomenclatura de las Unidades Territoriales Estadísticas (NUTS en francés), que es un estándar para demarcaciones territoriales empleado con fines estadísticos por la Unión Europea. Dicho estándar fue otorgado un estatus legal en el año 2003 y ha sido actualizado regularmente desde entonces. Actualmente se utiliza la versión NUTS 2016, pero ya está preparada la versión NUTS 2021, que incluye las implicaciones del Brexit y entrará en vigor el 1 de enero de 2021 (European Commission, 2018).

A día de hoy, desde su sede en Luxemburgo, Eurostat sigue promoviendo la estandarización internacional mediante la harmonización de métodos estadísticos en los países miembros de la Unión Europea (al igual que en sus candidatos de entrada) y en los miembros de la Asociación Europea de Libre Comercio.

Oficialmente, «la misión de Eurostat es proporcionar estadísticas de alta calidad para Europa» (Eurostat, 2018). Cumple dicha misión al poner al alcance de todos información amplia y detallada que resulta de interés para gobiernos, negocios, el sector educativo, periodistas y el público general. Los datos se pueden visualizar en la propia página web o se pueden extraer desde numerosos entornos. Uno de ellos es a través del lenguaje de programación R.

Es más, para facilitar la comprensión de las estadísticas por los usuarios, Eurostat dispone de apartados completos como *Statistics Explained*, que aporta explicaciones y artículos de estadística detallados. La base de datos europea

también otorga disponibilidad completa a publicaciones, comunicados de prensa, novedades y herramientas de visualización; e incluso ofrece servicios de apoyo a usuarios, medios de comunicación y a instituciones (nacionales y estadísticas).

1.2 Estructura de la base de datos Eurostat

Eurostat ofrece más de 8200 conjuntos de datos, por lo que no seríamos capaces de listarlos uno a uno. Sin embargo, la página oficial nos sugiere una división general por temas, indicando los siguientes 9 grupos:

- Estadísticas generales y regionales
- Economía y finanzas
- Población y condiciones sociales
- Industria, comercio y servicios
- Agricultura y pesca
- Comercio internacional
- Transporte
- Medio ambiente y energía
- Ciencia, tecnología y sociedad digital

La página muestra como cada grupo contiene de 1 a 15 subgrupos, con el fin de poder explicar superficialmente los datos contenidos en cada grupo. Sin embargo, en realidad la base de datos cuenta con una clasificación en cascada mucho más detallada. La Figura 1 muestra el comienzo de la lista de conjuntos de datos, visto desde la página web de la base de datos:

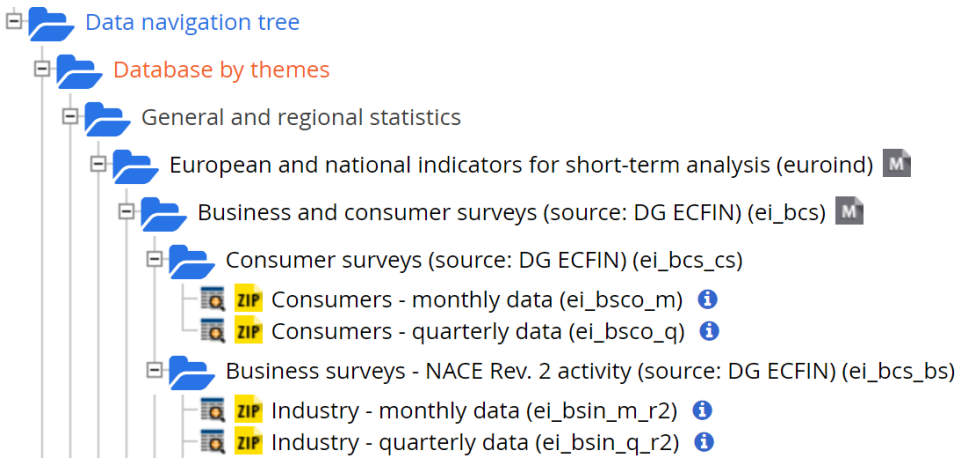

Figura 1: Muestra de la estructura de Eurostat

Como todas las bases de datos, Eurostat está compuesta por multitud de diversos conjuntos de datos. Concretamente, diferencia entre dos tipos: *Datasets* y *Tables*. Por simplificación, en este apartado nos referiremos a los primeros como «conjuntos de datos» y a los segundos como «tablas», y cuando hagamos referencia a alguno sin importar su tipo usaremos palabras como «ítem». Las diferencias entre ambos son estrechas, y en R se extraen y tratan de la misma forma.

Cada ítem se puede conseguir de diferentes formas, pero la más relevante a la hora de usar R es mediante su código de identificación. Explicaremos el procedimiento en el capítulo **3. Creación de informes con R Markdown**.

A continuación vamos a analizar el primer nivel de la estructura de Eurostat, tal y como viene presentada en la misma página web de la base de datos. Observamos que el árbol de navegación de datos está dividido en 6 grandes apartados:

- **Base de datos por temas**, la ordenación de la que hablábamos antes. Parece que es la predilecta, al menos para ayudar al usuario a comprender los datos que contiene Eurostat de forma general. Cada conjunto de datos puede visualizarse online mediante el explorador de datos de eurostat.
- **Tablas por temas**. Sigue la misma ordenación por temas que el apartado anterior, pero no contiene los mismos datos y es menos completo. Cada tabla se puede visualizar gracias a un navegador de datos (distinto del explorador del apartado previo) o a través de una interfaz de tablas, grafos y mapas.
- **Tablas según la política de la UE**. Este apartado presenta una tablas seleccionadas y organizadas siguiendo políticas específicas de la UE. Como se trata de tablas, se pueden visualizar mediante el navegador o la interfaz.

- **Temas transversales**. Aquí se pone a la vista un surtido de datos centrados en temas destacables, de actualidad. Dependiendo de si el ítem es un conjunto de datos o tabla, se puede visualizar mediante el explorador o mediante el navegador y interfaz.
- **Entradas nuevas**. En este apartado se resaltan las inclusiones a Eurostat más recientes.
- **Ítems recientemente actualizados**. Aquí se cataloga sencillamente una lista ordenada por fecha de actualización y código.

Destacamos que cada uno de los ítems, ya sea conjunto de datos o tabla, siempre se puede descargar como un archivo .tsv comprimido (*Tab-separated values* o valores separados por tabuladores), que es parecido al .csv pero utiliza tabulaciones en lugar de comas. Además, cada ítem dispone siempre de información sobre el mismo y ocasionalmente de una página explicando los metadatos en la que podemos encontrar, por ejemplo, la explicación y método de cálculo de las unidades de medida empleadas.

Las diferencias entre el apartado de **Base de datos** y el de **Tablas** la encontramos explicada en la página de preguntas más frecuentes (European Commission, 2020a):

El apartado de **Base de datos** contiene el espectro completo de los datos puestos públicamente a disposición por Eurostat. Se presentan en tablas multidimensionales con numerosas opciones de selección y formatos de exportación.

Por otro lado, el apartado de **Tablas** únicamente ofrece una selección de los datos más importantes de Eurostat. Además, los presenta en una forma fácil de usar, mediante tablas de dos o tres dimensiones.

Cada uno de los apartados anteriores se divide en temas que a su vez se dividen en hasta otros 3 niveles de detalle consecutivos para finalmente llegar al ítem. Es más, muchos ítems contienen los mismos datos, pero algunos son estudios mensuales y otros trimestrales, cuatrimestrales etc. Es por ello que, aunque en realidad sea mucho más extensa, vamos a finalizar aquí el estudio de la estructura.

Tras estudiar Eurostat y extraer numerosos conjuntos de datos, hemos llegado a la conclusión de que utilizar el árbol de navegación es incómodo y se puede volver confuso. Por lo tanto, concluimos que lo más importante a la hora de buscar datos es:

1. Decidir el ámbito o tema del que se quiere extraer información y

2. Buscar un conjunto de datos adecuado para resolver el problema en cuestión o para arrojar luz a la toma de decisiones. También se recomienda seleccionar el conjunto de datos en base a la adecuación de sus variables o indicadores, la facilidad de interpretación de las unidades que presenta o incluso por su fecha o periodicidad de actualización.

2. Metodología

El objetivo de este capítulo es dar luz a las herramientas y métodos estadísticos de mayor importancia en el desarrollo de este proyecto. Concretamente, para cada herramienta daremos el contexto de su creación, contaremos brevemente su historia, explicaremos su utilidad general y aclararemos el uso concreto que le hemos dado.

2.1 Lenguaje de programación R

R (R Core Team, 2020) nace como una implementación del lenguaje de programación estadístico S. Fue creado por Ross Ihaka y Robert Gentleman en la Universidad de Auckland, Nueva Zelanda. Su desarrollo lo llevó a cabo el *R Development Core Team* desde 1992 hasta su primera versión Beta estable en Febrero del año 2000. La Figura 2 muestra el logo de R desde 2016.

R es un lenguaje y entorno diseñado para computación estadística y gráfica. Una de sus mayores ventajas es su facilidad de extensión mediante *packages* o paquetes (como el paquete «eurostat» del que hablaremos en el subapartado 2.3 Paquete eurostat para R). Se encuentra disponible como software libre para una variedad de plataformas Unix, Windows y MacOS.

Actualmente se beneficia del apoyo de *The R Foundation*, cuyas funciones incluyen llevar su mantenimiento; servir como punto de referencia y contacto para individuos, instituciones o empresas que buscan apoyar o interactuar con la comunidad de desarrollo; y administrar el copyright del software y de la documentación de R.

Todo el software y documentación mencionados, incluidos los paquetes, se encuentran completamente disponibles para descarga en la red mundial de servidores llamada CRAN (*the Comprehensive R Archive Network*).

Figura 2: Logo de R

2.2 Entorno de programación RStudio

Fundado en 2009 por J.J. Allaire, RStudio (RStudio Team, 2019) nace con el objetivo de crear software gratuito y de código abierto para la ciencia de datos, investigación científica y comunicación técnica. La organización promueve la producción y adquisición de conocimiento por todos, además de facilitar la colaboración y la investigación reproducible con el fin de mejorar la integridad y eficacia de los trabajos en ciencia, educación, gobierno e industria.

Uno de sus mayores logros es la creación y mantenimiento del entorno de desarrollo integrado (IDE en inglés) también llamado RStudio, que se ha convertido en uno de los mejores entornos estadísticos para el lenguaje de programación R. El IDE permite el acceso modular a diversidad de funciones, como una consola, un editor de sintaxis y numerosas herramientas para el trazado, depuración y gestión del espacio de trabajo.

El equipo RStudio también es responsable de la creación de diversos paquetes y extensiones para el lenguaje de programación R. En este proyecto hemos utilizado algunos de ellos, como el paquete «tidyverse» (Wickham et al., 2019), que incluye una multitud de funciones y facilidades para la ciencia de datos; o el paquete «shiny» (Chang, Cheng, Allaire, Xie, y McPherson, 2020), que permite la creación de aplicaciones interactivas.

La Figura 3 presenta uno de los logos que Rstudio pone a disposición de los usuarios, siempre y cuando sigan las pautas de uso.

Figura 3: Logo de RStudio

2.3 Paquete eurostat para R

Como ya hemos mencionado en el apartado 1.2 Estructura de la base de datos Eurostat, existen diversas formas de acceder a los contenidos de la base de datos europea. La primera opción suele ser descargar los conjuntos de datos de la página web, pero la facilidad de extensión de R nos permite acceder directamente

desde el propio lenguaje de programación, facilitando el dinamismo y actualización.

Creado por Leo Lahti, Janne Huovari, Markus Kainu y Przemyslaw Biecek, el paquete «eurostat» (Lahti, Huovari, Kainu, y Biecek, 2017) añade una multitud de funciones a R para poder descargar y manipular los datos de Eurostat. Como todo el software disponible para R, el paquete se puede descargar desde CRAN de forma gratuita o incluso desde el propio lenguaje mediante el comando «install.packages("eurostat")».

En este trabajo hemos utilizado el paquete principalmente para visualizar el índice de conjuntos de datos, buscar datos por nombre y descargarlos por su código. Los detalles de uso los explicaremos en el siguiente capítulo: **3. Creación de informes con R Markdown**.

En la Figura 4 podemos ver uno de los logos de Eurostat, que siempre se encuentra visible cuando navegamos por su página web.

Figura 4: Logo de Eurostat

2.4 Series temporales y modelos ARIMA

Eurostat contiene conjuntos de datos que muestran la evolución de variables o indicadores a lo largo del tiempo. Por lo tanto, podemos considerar esos conjuntos como series temporales, que vamos a definir a continuación. En estadística, se denomina serie temporal a una «secuencia de valores o lecturas ordenados por un parámetro temporal» (Granger, Newbold, y Shell, 2014). Según el análisis clásico, toda serie temporal se puede dividir en 4 componentes (Chirivella, 2018):

- **Tendencia**: Trayectoria a largo plazo de la serie, indica si la variable de estudio crece o disminuye con el tiempo.
- **Estacionalidad**: Oscilaciones periódicas de frecuencia inferior a un año. Son causadas por tendencias culturales o climáticas, como por ejemplo que aumente el turismo en España cuando llega el verano.
- **Variación cíclica**: Oscilaciones periódicas de frecuencia superior a un año. Pueden deberse a macro tendencias culturales y suele ser difícil observarlas

porque el rango de fechas de los datos puede no ser lo suficientemente amplio como para poder visualizar un ciclo.
- **Variación irregular o residual**: Variabilidad en el comportamiento de la serie por pequeñas causas impredecibles. Aquí se agrupan los movimientos que no describen las demás componentes.

Para analizar y predecir series temporales utilizaremos modelos Autorregresivos Integrados de Media Móvil (ARIMA). En esencia, estos modelos realizan estimaciones futuras en base a datos del pasado, en lugar de basarse en variables independientes. Para un estudio detallado de series temporales se recomienda la lectura del texto de Brockwell y Davis (2002).

Para poder aplicar un modelo ARIMA sobre una serie temporal, los datos deben cumplir dos condiciones:

- Ser estacionarios, es decir que las propiedades de la serie no dependan del momento en que se capturan.
- Ser univariantes, o sea de una sola variable.

De forma simplificada, el modelo ARIMA se ajusta a los datos mediante tres parámetros: p, d y q. Estos son enteros positivos que hacen referencia respectivamente al orden de las siguientes partes del modelo: medias autorregresivas, integradas y móviles.

En el caso de que se trate de un modelo con tendencia y estacionalidad, el modelo resultante tendrá un total de seis parámetros. Se corresponden con los tres anteriores, pero para la parte de tendencia y la parte de estacionalidad. En ese caso, el modelo se definirá como ARIMA(p,d,q)(P,D,Q)s , donde «s» corresponde al periodo de estacionalidad.

Para encontrar el orden del modelo utilizaremos un paquete de R llamado «forecast» (Hyndman et al., 2020), recurriendo a la función «auto.arima()», que utiliza una variación del algoritmo Hyndman-Khandakar (Hyndman y Khandakar, 2008). Concretamente, emplea una conjunción de pruebas de raíz unitaria, minimización del criterio de información de Akaike (AIC en inglés) y estimación de máxima verosimilitud (MLE en inglés).

El criterio definitivo es el AIC, que se fundamenta en el principio de parsimonia (también conocido como navaja de Ockham). El principio dicta que «en igualdad de condiciones la explicación que es más simple suele ser la verdadera» (DeSignificado.com, 2020). Como explica Date (2020), el criterio AIC recompensa

modelos con una elevada bondad del ajuste (corta distancia entre los valores generados por el modelo y los valores observados en la serie temporal) y penaliza aquellos con una elevada complejidad. Concretamente, la fórmula para calcular la valoración AIC se basa en una sustracción entre el número de parámetros del modelo y una medida de la bondad del ajuste.

2.5 Procedimiento de modelado ARIMA

En este apartado vamos a explicar el procedimiento para generar un modelo ARIMA y efectuar una predicción. Nuestra guía ha sido el libro de Hyndman y Athanasopoulos (2018), de donde extraemos la Figura 5, que resume todo el procedimiento. También nos hemos inspirado del ejemplo de Campos (2018), que explica como realizar cada paso.

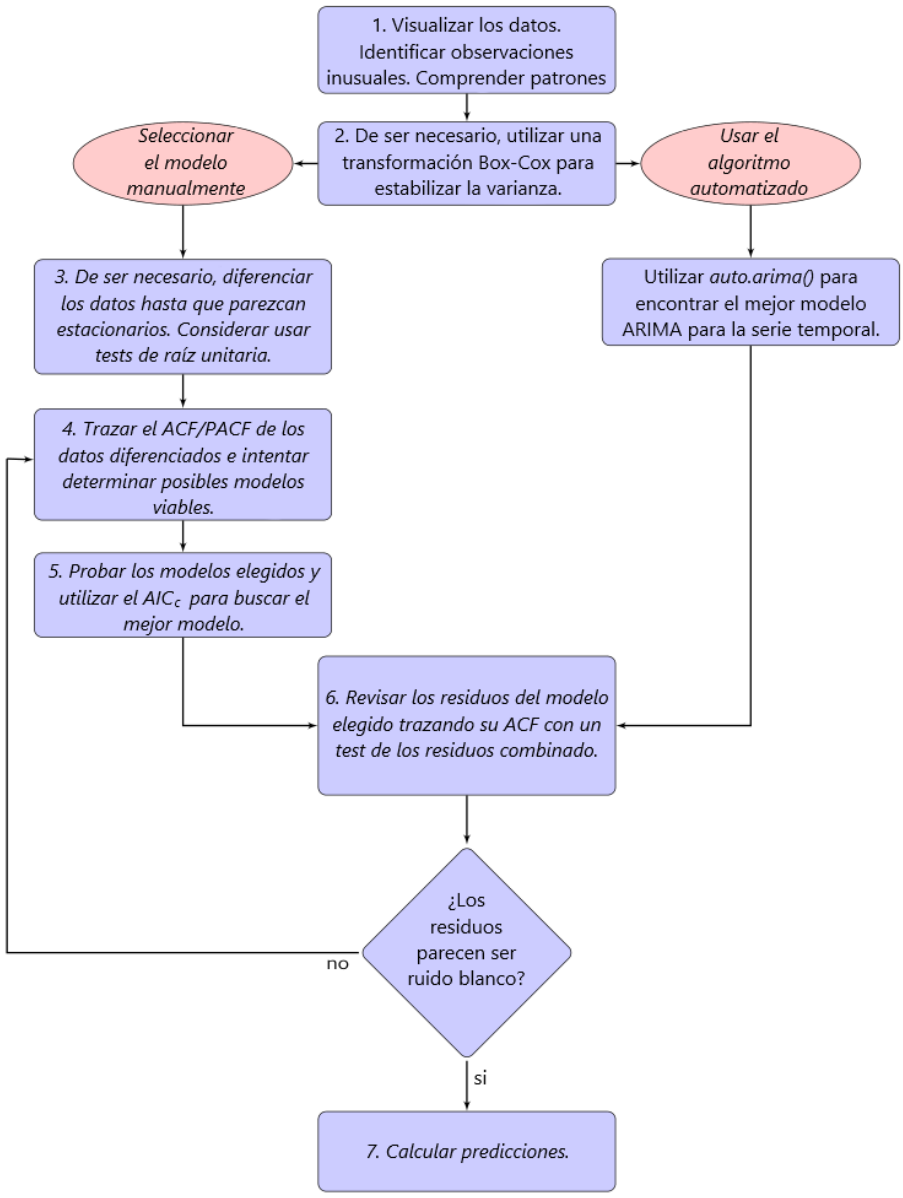

Figura 5: Procedimiento general de modelado ARIMA en R

A continuación detallaremos cada uno de los pasos, aportando las explicaciones necesarias:

Paso 1: Trazar los datos. Es importante que los datos sean univariantes y se presenten en periodos inferiores a un año, de lo contrario no se podrá observar estacionalidad (recordamos que se da en frecuencias inferiores al año) y el modelado será impreciso. Esta es la razón por la que aplicaremos este procedimiento de forma práctica en el apartado 5.4 Modelado ARIMA y predicción: PIB, ya que el

dataset de Eurostat «PIB y principales componentes (resultado, gasto e ingreso)» tiene una periodicidad trimestral. Este paso nos permite comprender la tendencia de la serie y asegurarnos de que no existen patrones inesperados.

Paso 2: Utilizar una transformación Box-Cox si fuera necesario. La varianza es «una medida de dispersión que representa la variabilidad de una serie de datos respecto a su media» (López, 2019). En el caso de que la varianza de la serie no esté estabilizada, se deben realizar una serie de transformaciones para «corregir sesgos en la distribución de errores, corregir varianzas desiguales y corregir la no linealidad en la relación (mejorar la correlación entre las variables)» (Kutner, 2005). Una vez que todo está en orden, se deben convertir los datos en una serie temporal.

Paso 3: Ajustar el modelo. Habitualmente aquí el procedimiento incluiría asegurar que los datos sean estacionarios, realizar pruebas ACF/PACF para extraer posibles modelos y buscar el mejor mediante el criterio AIC. La función de autocorrelación (ACF en inglés, FAS en español) y la función de autocorrelación parcial (PACF en inglés, FAP en español) permiten visualizar las «medidas de asociación entre valores de series actuales y pasadas e indican cuáles son los valores de series pasadas más útiles para predecir valores futuros» (IBM Knowledge Center, 2020). Son especialmente útiles para determinar el orden de un modelo ARIMA.

Afortunadamente, la función «auto.arima()» se encarga de todos estos pasos y nos proporciona directamente el modelo con el AIC más bajo.

Paso 4: Comprobar que los residuos son ruido blanco. Los residuos son la diferencia entre los valores observados (en la serie temporal) y los valores predichos (por el modelo). Para ello realizaremos lo que se conoce como la prueba de Box-Ljung, que permite comprobar si existe un patrón entre los residuos, ya que en ese caso deberían tenerse en cuenta en el modelo. Efectuaremos la prueba en tres pasos:

- Un ACF de los residuos, que nos permitirá evaluar su autocorrelación. Lo deseable es que no encontremos autocorrelaciones significativas.
- Una prueba de Ljung-Box Q, que sirve para observar posibles dependencias en los datos. Idealmente no constataremos dependencias.
- Un gráfico Q-Q, cuya utilidad es observar si los residuos siguen una distribución normal. Si sus valores siguen la línea, los consideraremos normales, lo que sería idóneo.

En base a los resultados, si consideramos que no existe un patrón en los residuos, los podremos valorar como ruido blanco. En el caso contrario, deberemos volver al **Paso 2** y aplicar transformaciones.

Paso 5: Calcular predicciones. Finalmente podemos utilizar el modelo ARIMA obtenido para aplicarle la función «forecast» del paquete con su mismo nombre. De esa forma obtendremos una proyección en forma de serie de datos, que podremos visualizar mediante un gráfico.

3. Creación de informes con R Markdown

En este capítulo explicaremos los aspectos más importantes de la creación de la aplicación y los informes. Destacamos la importancia del paquete «tidyverse» (Wickham et al., 2019), que ha sido de mucha utilidad para el tratamiento de datos, gracias a funciones como las *pipelines* (permiten encadenar transformaciones, se escriben «%>%») y para su representación, gracias a funciones como «ggplot2» (Wickham, 2016) que permiten dibujar gráficas avanzadas.

3.1 Aspectos generales

En este apartado expondremos los rasgos universales de mayor relevancia a la hora del desarrollo y del funcionamiento de la aplicación.

Selección de datos

De forma general, el procedimiento que hemos seguido para seleccionar cada conjunto de datos ha sido:

1. Entrar en Rstudio y cargar las librerías «eurostat» y «tidyverse».
2. Buscar usando el índice (mediante «get_eurostat_toc()») o buscando por palabra clave (mediante «search_eurostat()»).
3. Seleccionar un conjunto de datos (*dataset*) interesante. Un criterio importante es la fecha de los datos más recientes, ya que cuanto más nueva sea mejor refleja la capacidad de actualización automática de los informes. Al descargarlo activamos la opción «type = label» para trabajar con nombres de variables inteligibles.
4. Elegir qué variables y datos mostrar, observando las distintas opciones con «distinct()». Cada conjunto de datos puede tener más de 10 variables mostradas en más de 10 unidades distintas, dando un total de 5 millones de filas. Es importante escoger unidades homogéneas y datos que se puedan representar adecuadamente.
5. Depurar, filtrar y renombrar variables y datos para que el conjunto de datos resultante sea comprensible y se adapte bien a los formatos de tabla y gráfico.
6. Elegir el formato de representación y las acciones que podrá realizar el usuario.

A continuación explicaremos brevemente el procedimiento de creación de la aplicación y de cada uno de los informes.

Aplicación e interactividad

Una de las premisas del trabajo fue la siguiente: El objetivo es poder visualizar datos actualizados extraídos de Eurostat, de conjuntos de datos seleccionados de antemano. Este factor nos permite ir más allá de la creación de informes estáticos, como sería un pdf, por lo que decidimos que la visualización debería ser interactiva.

De esta forma el usuario tendrá toda la información a su disposición y además podrá elegir qué datos visualizar mediante el filtrado de fechas y variables. El objetivo es que la navegación e interpretación de los datos sea más dinámica y entretenida, a diferencia de los habituales informes sobre papel.

La interactividad la hemos logrado gracias al paquete «shiny» (Chang et al., 2020). Dicha extensión de R contiene numerosos *Widgets*, que son selectores de diversos tipos. Algunos ejemplos son: *Select box*, que permite seleccionar una opción de una lista; *Checkbox group*, cuya utilidad reside en poder tamizar una o varias opciones simultáneamente; y *Slider*, que sirve para elegir uno o dos valores dentro de un rango y puede ser utilizado para fechas.

El archivo sigue una estructura interfaz-servidor en la que la interfaz contiene los elementos visuales, incluídos los *Widgets*; y el servidor todo lo que es ejecución dinámica, incluído el tratamiento de datos. Anteriormente la interfaz y el servidor debían estar en distintos archivos, pero ahora ya pueden ubicarse en el mismo.

Finalmente, por cuestiones de estilo y presentación, hemos utilizado el paquete «shinydashboard» (Chang y Borges Ribeiro, 2018) que funciona como una extensión de Shiny. Lo hemos empleado principalmente para crear un sistema de cajas con código de colores, una cabecera y una navegación por pestañas, mostrando la estructura final del trabajo como en la Figura 6:

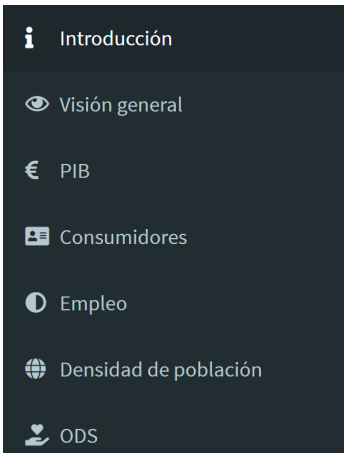

Figura 6: Pestañas de la aplicación

3.2 Informes dinámicos

En este apartado explicaremos superficialmente el proceso de desarrollo y exposición de los datos a través de los informes dinámicos, cuya estructura se puede observar en la Figura 6.

Pestaña de «Introducción»

Esta primera pestaña tiene el propósito de suavizar el primer contacto entre el usuario y la aplicación. Para ello se trata únicamente de una portada introductoria, sencilla y estática.

En la aplicación, el aspecto que tiene se puede ver en la Figura 7:

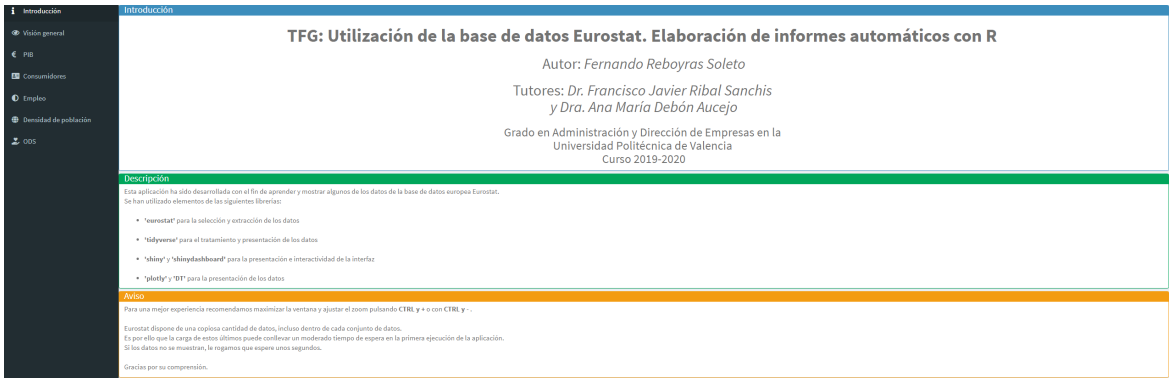

Figura 7: Pestaña de 'Introducción'

Pestaña de «Visión General»

Siguiendo en la línea de familiarizar al usuario con los datos, esta pestaña presenta de forma resumida algunos de los datos que se pueden encontrar en los informes. Para suavizar la presentación hemos utilizado el formato más básico: una tabla de datos.

Sin embargo, la tabla de Shiny es básica, y por lo tanto carece de opciones de visualización. En consecuencia, hemos recurrido al paquete «DT», que otorga acceso a *Datatables* (Yihui, 2017), que son tablas parecidas a las básicas, pero permiten una presentación por páginas, ordenar las columnas y el uso de una barra de búsqueda.

La apariencia de esta pestaña se muestra en la Figura 8:

Figura 8: Pestaña de 'Visión General'

Pestaña de «PIB»

En esta pestaña se accede al conjunto de datos llamado «*GDP and main components (output, expenditure and income)*», que en español es «PIB y principales componentes (resultado, gasto e ingreso)». Su código Eurostat es «namq_10_gdp» y aquí se encuentra la página web del *dataset*. En esencia, este conjunto de datos muestra los valores de ciertos indicadores económicos nacionales como el Producto Interior Bruto (PIB), el Gasto en Consumo Final o la Formación Bruta de Capital Fijo para una multitud de países de la Unión Europea y de la Zona del Euro.

Una transformación que hemos realizado es filtrar la unidad según la fecha. Hemos decidido utilizar los volúmenes encadenados en millones de euros con base 2015 para los datos a partir del 2015, con base 2010 para aquellos entre 2005 y 2010, y con base 2005 para el resto (2005, 2010 y 2015 eran las tres bases disponibles). También había multitud de otras unidades, como en unidades de moneda nacional o índices base 100, pero hemos considerado que utilizar euros permitiría una comparación más intuitiva, al tratarse de la moneda oficial de la UE.

El conjunto de datos en cuestión es una serie temporal, que como explicamos en el apartado 2.4 Series temporales y modelos ARIMA, tiene un componente de estacionalidad. En este ejemplo, la serie temporal marcaría la evolución del PIB a través de los años. Sin embargo, se sabe que la economía de cada país no es constante a lo largo del año, sino que tiene temporadas altas y bajas, dependiendo de sus principales sectores. Tomando el ejemplo de España, el turismo es de los sectores más influyentes, pero genera un mayor crecimiento del PIB en verano.

En este *dataset*, los datos son tomados para cada trimestre. Por lo tanto, con el objetivo de comparar correctamente dos trimestres entre sí, lo correcto es eliminar el efecto de la estacionalidad para poder observar las variaciones causadas por factores relevantes, no por el simple hecho de pertenecer a momentos del año distintos. Sin embargo, hay que tener en cuenta que este tipo de medidas pueden acentuar el impacto de la reciente pandemia causada por el COVID-19, ya que los efectos estacionales se han visto alterados de manera inesperada.

Con el fin de mejorar la calidad del estudio y simplificar la visualización, hemos decidido eliminar la estacionalidad de la serie, o según Eurostat, «estimar y eliminar los efectos de las estaciones de las series temporales para hacer que desaparezcan las fluctuaciones estacionales» (Eurostat, 2020). Para ello hemos filtrado los datos conservando aquellos ajustados por estación y calendario, ya que es una opción que viene directamente en el *dataset*.

Finalmente, con el objetivo de mostrar el estado de la UE en cada instante de tiempo, hemos tenido en cuenta las fechas de entrada o salida de países de la UE. Pongamos un ejemplo: los datos vienen con 2 variables distintas, UE con 28 países (2013-2020) y UE con 27 países (desde 2020). Se trata de la salida de Reino Unido, que a nivel estadístico entró en vigor el 1 de febrero 2020 (European Commission, 2020b). Por lo tanto, con el fin de representar de forma fiel la Unión Europea en cada instante de tiempo, aplicaremos la variable con 28 países hasta el 1 de febrero de 2020 (excluído) y la variable con 27 países a partir de ese mismo día.

En futuros conjuntos de datos volveremos a realizar estas dos últimas transformaciones, que denominaremos como **eliminación de estacionalidad** y **ajuste de representación fiel de la UE**.

Aunque Eurostat muestre datos para más de 30 países, hemos seleccionado los que consideramos interesantes para comparar con España y mostrar el rango que tiene la base de datos: la Unión Europea, la Zona del Euro, España, Francia, Alemania y Reino Unido.

Esta pestaña se puede observar en la Figura 9:

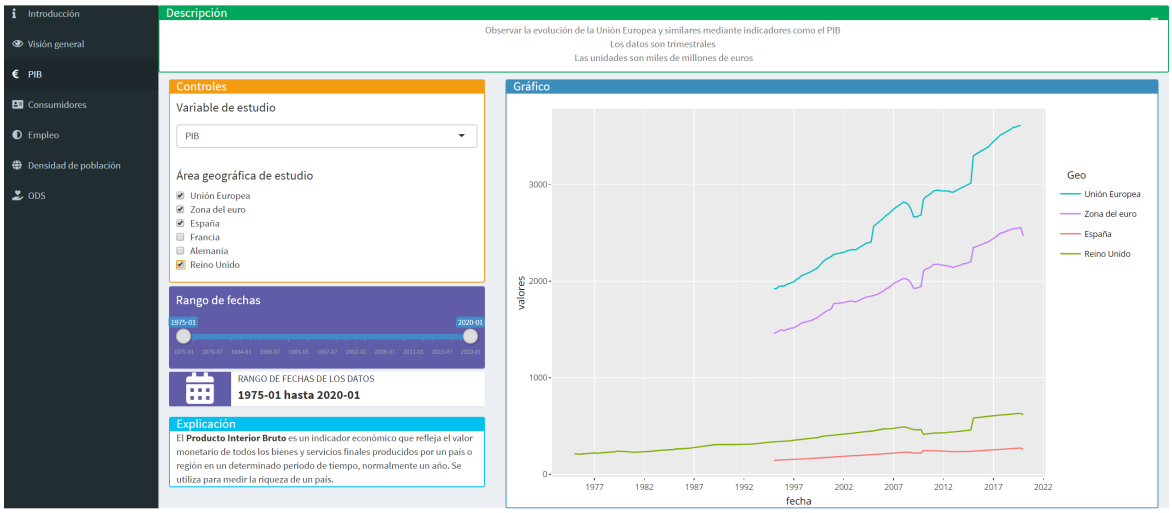

Figura 9: Pestaña de ’PIB’

Destacamos que hemos realizado un modelado ARIMA y predicción sobre este conjunto de datos en el apartado 4.4 Modelado ARIMA y predicción: PIB.

Pestaña de «Consumidores»

Aquí el conjunto de datos en cuestión es «*Consumers - monthly data*» («Consumidores - datos mensuales»), de código «ei_bsco_m» y con la siguiente página web del *dataset*. Las variables son indicadores medidos a partir del consumidor habitual, como el «Indicador de la confianza de los consumidores», y previsiones como su «Situación financiera para los próximos 12 meses» o su «Situación económica general para los próximos 12 meses». Hemos seleccionado aquellos que a nuestro parecer representan mejor el conjunto de datos y son relativamente fáciles de comprender.

De la misma forma que con el *dataset* previo, aplicamos la eliminación de estacionalidad, aunque aquí no realiza ajustes de calendario, solo de estación.

El rango de países es similar al conjunto de datos anterior, pero para simplificar la visualización y hacerla más variada hemos decidido mostrar únicamente los datos de la Unión Europea, permitiendo en su lugar visualizar distintos indicadores simultáneamente.

En la práctica, esta pestaña se presenta en la Figura 10:

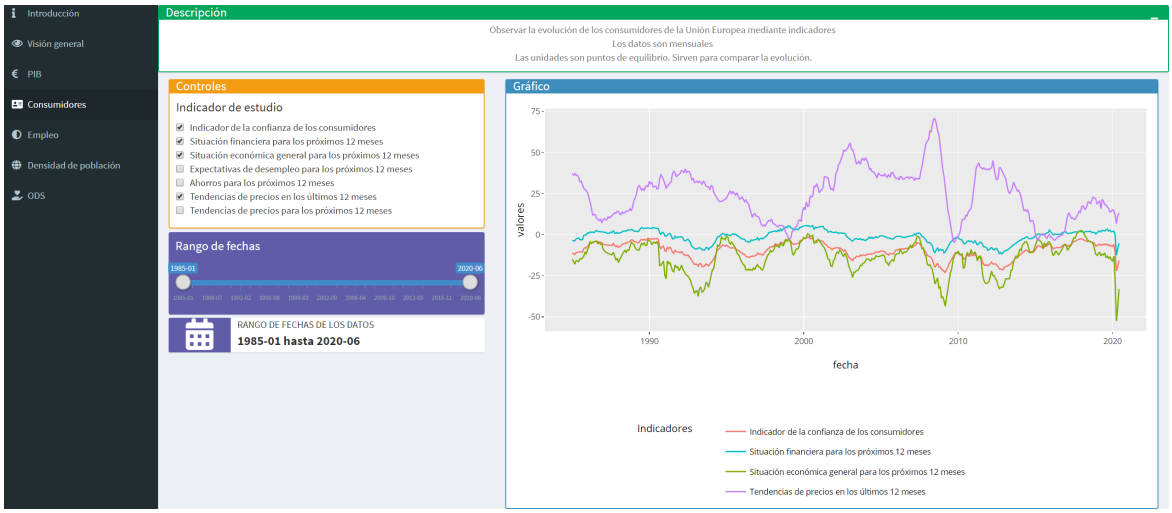

Figura 10: Pestaña de ’Consumidores’

Pestaña de «Empleo»

Para esta pestaña hemos buscado conjuntos de datos relacionados con el desempleo en la Unión Europea. Afortunadamente, Eurostat dispone no solo de información sobre el desempleo, sino también sobre las vacantes de empleo. El primer *dataset* se llama «*Harmonised unemployment rates* (%) - *monthly data*» («Ratios de desempleo harmonizados (%) - datos mensuales»). Su código es «ei_lmhr_m» y esta es la página web del *dataset* sobre desempleo. El segundo conjunto de datos tiene el nombre de «*Job vacancy rate*» («Tasa de vacantes de empleo»), el código «ei_lmjv_q_r2» y esta es la página web del *dataset* sobre vacantes.

Respecto al conjunto de datos sobre el desempleo, se puede visualizar el ratio de desempleo total, por género, por edad (distinguiendo entre mayores y menores de 25 años), y por género y edad simultáneamente. El tratamiento que le damos incluye eliminación de estacionalidad (sin ajustes de calendario) y el ajuste de representación fiel de la UE.

Por otra parte, el *dataset* del ratio de vacantes de empleo nos permite comparar dicho ratio entre distintos sectores como el de la «Economía empresarial», de la «Información y comunicación» o el «Científico y técnico». Sin embargo, nos vemos forzados a escoger datos sin ajuste alguno de estacionalidad, ya que los datos con ajuste filtrados para la Unión Europea apenas se recogen para el sector de «Economía de Negocios» y el sector de «Industria, Construcción y Servicios». También realizamos el ajuste de representación fiel de la UE.

Ambos *datasets* se pueden visualizar en la misma pestaña, compartiendo controles como muestra la Figura 11:

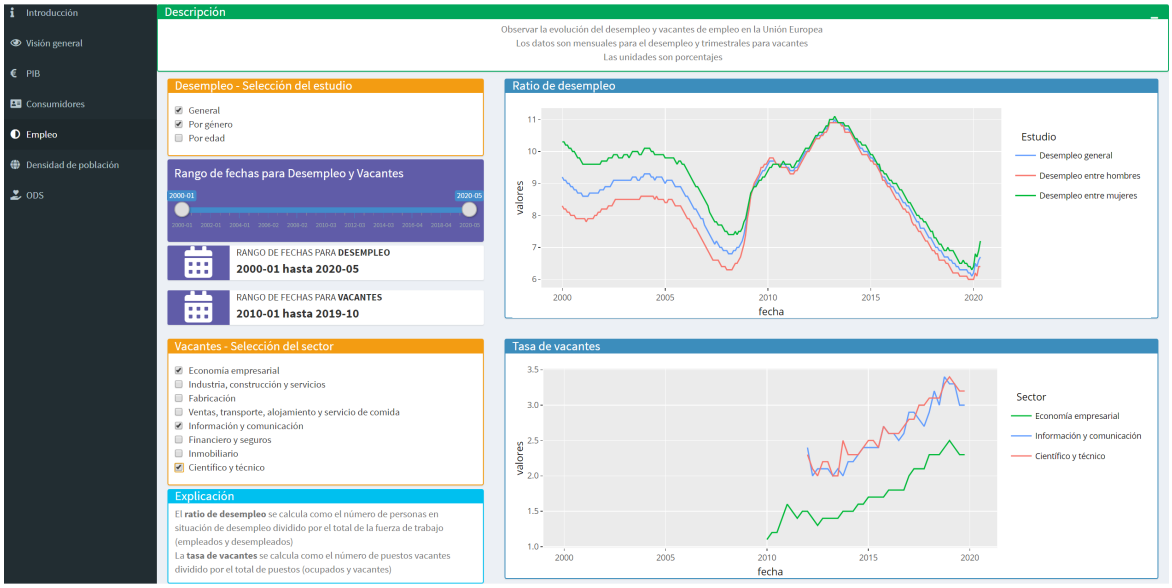

Figura 11: Pestaña de ’Empleo’

Pestaña de «Densidad de población»

El objetivo último de esta pestaña es mostrar un ejemplo real de la NUTS, que explicamos en el apartado 1.1 Origen y evolución de Eurostat, aplicada al caso de España. Para ello hemos buscado datos que muestren una precisión de hasta NUTS 3 y escogimos uno de los más habituales y fáciles de comprender: la densidad de población. El nombre del conjunto de datos es «*Population density by NUTS 3 region*» («Densidad de población por región NUTS 3»), su código es «demo_r_d3dens» y esta es la página web del *dataset*.

Originalmente nuestra idea era mostrar el mapa de España con distintas demarcaciones según el nivel NUTS seleccionado, pero tras varios intentos tuvimos que desistir. El principal motivo es que para generar un mapa en R es necesario cargar las coordenadas de las fronteras, lo que requiere de un largo tiempo de cómputo.

Dado que la carga de los *datasets* ya es extensa y que la idea es que la aplicación sea ejecutada cada vez que se quiera consultar los datos, en lugar de una sola vez, descartamos la idea de generar un mapa dinámicamente. De haberlo incluido, el tiempo de carga de la aplicación sería de hasta 3 minutos más largo.

En su lugar hemos representado los datos mediante una tabla que permite ver para cada región su código NUTS, su nombre y su densidad de población.

En lo que concierne al tratamiento del conjunto de datos, ha sido bastante escueto. Tan solo hemos filtrado geográficamente para mostrar los datos de España y hemos hecho una separación y unión para que en la misma tabla aparezcan el cógido NUTS de cada región y su nombre en español (por ejemplo el código de la ciudad de Valencia es «ES523»).

El resultado final de la pestaña se puede ver a en la Figura 12:

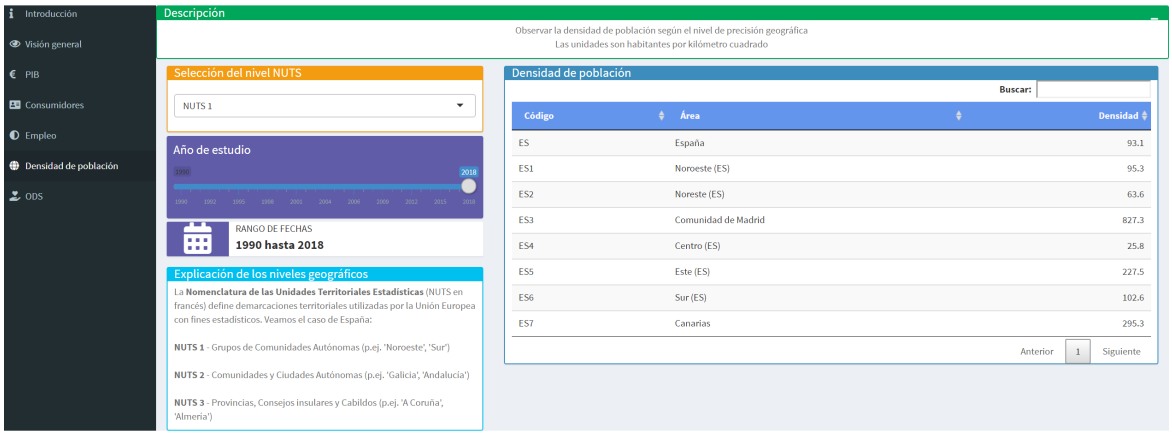

Figura 12: Pestaña de ’Densidad de población’

Pestaña de «ODS»

Finalmente, nos gustaría resaltar la importancia del desarrollo sostenible, especialmente en los ámbitos económicos y sociales. Cada uno de nosotros puede aportar su pequeño grano de arena para hacer de nuestro mundo un lugar mejor, pero quienes realmente tienen el poder de hacer cambios importantes son las grandes multinacionales y los gobiernos.

En 2015, los líderes mundiales establecieron una serie de objetivos para «erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos» (Naciones Unidas, 2020). Es por ello que nos hemos propuesto incluir un apartado sobre el cumplimiento de los ODS propuestos por la ONU. Para la creación de esta pestaña hemos seguido un procedimiento ligeramente distinto, por lo que vamos a explicarlo detalladamente en el capítulo siguiente: **4. Ejemplos detallados**.

4. Ejemplos detallados

En este capítulo vamos a explicar paso por paso el procedimiento de creación de la última pestaña, que trata sobre Objetivos de Desarrollo Sostenible. De esta forma expondremos claramente el modo de uso de la base de datos Eurostat mediante el lenguaje de programación R. Adicionalmente, resaltaremos las diferencias y mejoras que conlleva el uso de informes interactivos respecto a los habituales informes estáticos y constataremos brevemente el cumplimiento de los ODS por parte de la Unión Europea y de España.

Por añaditura, mostraremos un ejemplo de utilidad posterior de los datos que ofrece Eurostat utilizando uno de sus *datasets* para crear un modelo ARIMA con en fin de realizar predicciones sobre los próximos años.

El primer paso es seleccionar los datos. Una vez tomada la decisión de que tratarán sobre los ODS, solo queda ver cómo Eurostat los mide y almacena. Como ya hemos explicado en el apartado 1.2 Estructura de la base de datos Eurostat, uno de los principales apartados que se pueden observar en la presentación de la base de datos es **Tablas según la política de la UE**. Es más, uno de sus principales componentes se llama **Indicadores de desarrollo sostenible** y contiene carpetas para todos y cada uno de los 17 ODS como se puede ver en la Figura 13:

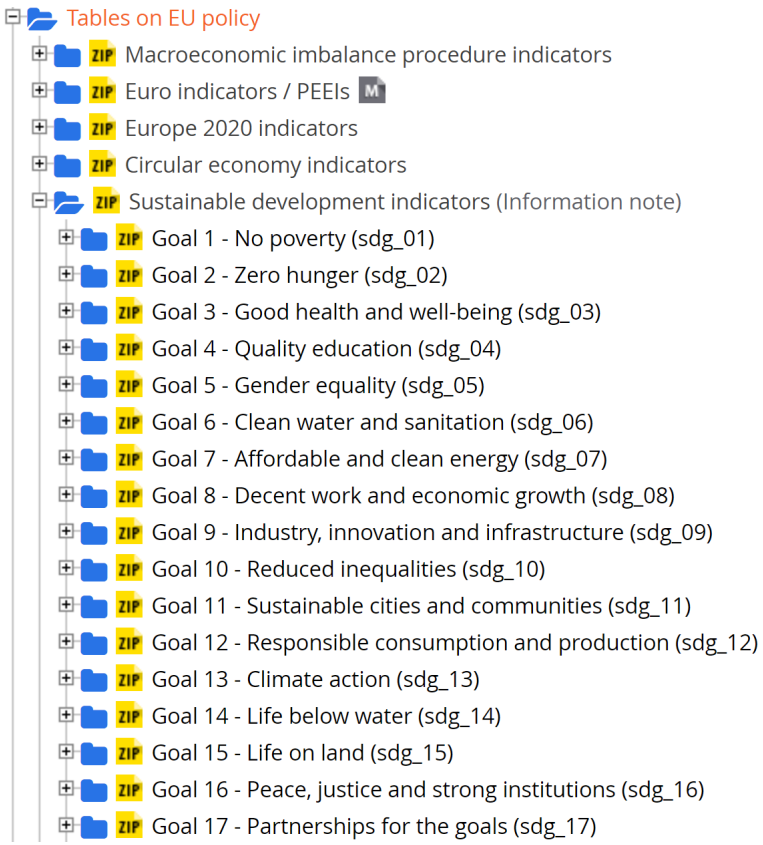

Figura 13: Eurostat: Indicadores de desarrollo sostenible

Destacamos que cada una de las carpetas de ODS contiene hasta 12 *datasets* distintos para medir el cumplimiento de dicho ODS. Sin embargo, como ya hemos mencionado numerosas veces, no nos vemos capacitados para mostrar la inmensidad de datos que contiene Eurostat, así que nos limitaremos a seleccionar dos conjuntos de datos de un objetivo distinto cada uno.

Tras leer y estudiar cada uno de los ODS tal y como se presentan en Eurostat, nos hemos decantado por *datasets* pertenecientes a los objetivos «Objetivo 5 - Igualdad de género» y al «Objetivo 13 - Acción por el clima». Ambos ODS están a la orden del día y son de los que más visibilidad mediática disponen. Además, estimamos que son objetivos en los que España tiene amplia capacidad de mejora, afirmación que solo podremos confirmar mediante su estudio.

A continuación explicaremos el procedimiento seguido para cada conjunto de datos.

4.1 Brecha de género en la contratación

Los primeros datos provienen de la tabla llamada «*Gender Employment Gap*» («Brecha de Empleo en la Contratación»), de código «sdg_05_30» y con la siguiente página web.

La medida se realiza mediante un indicador, que mide la diferencia entre los ratios de contratación de hombres y mujeres de edad entre 20 y 64 años. El ratio de contratación se calcula dividiendo el número de personas que trabajan y se encuentran en el rango de edad indicado entre la población total del mismo rango de edad. Este indicador se basa en la Encuesta de Fuerza Laboral de la UE. Como podemos comprobar en los metadatos, la unidad de medida es en puntos porcentuales y la periodicidad de los datos es anual.

Una vez estudiados los datos, procedemos a su transformación. Empezamos importando los datos, utilizando la función «get_eurostat()» cambiando el parámetro «type» a «label» para que datos que se verían como «PC_POP» se lean como «Percentage of total population» y se puedan interpretar sin recurrir a la documentación.

```
gender_data <- get_eurostat("sdg_05_30", type = "label")
```

A continuación comprobamos mediante «distinct()» los distintos valores que adopta cada columna. Constatamos que las columnas «age», «unit» y «indic_em» tienen un único valor, por lo que las excluimos seleccionando las restantes, pero guardamos la información que contienen para su interpretación.

```
gender_data <- select(gender_data, geo, time, values)
```

Luego realizamos el ajuste de representación fiel de la UE, de forma parecida a los ejemplos anteriores. En este conjunto de datos tenemos separación en 2020, por la salida de Reino Unido de la Unión Europea. Aprovechamos para filtrar escogiendo datos de las áreas geográficas deseadas, que para este ejemplo serán solo la UE y España.

```
gender_data <- filter(
  gender_data, time >= "2020-02-01"
                & geo == "European Union - 27 countries (from 2020)"
                          # Salida de Reino Unido
               | time < "2020-02-01"
                & geo == "European Union - 28 countries (2013-2020)"
               | geo %in% c("Spain"))
```

Finalmente solo quedan ajustes de presentación, como cambiar los nombres de las variables y datos a su versión en español. En la práctica, en vez de actualizar repetidamente la variable que contiene los datos, encadenamos las instrucciones mediante la *pipeline* (%>%) que nos proporciona el paquete «tidyverse». El resultado es el siguiente:

```
gender_data <- get_eurostat("sdg_05_30", type = "label") %>%
    select(geo, time, values) %>%
    filter(time >= "2020-02-01"
              & geo == "European Union - 27 countries (from 2020)"
                       # Salida de Reino Unido
           | time < "2020-02-01"
              & geo == "European Union - 28 countries (2013-2020)"
           | geo %in% c("Spain")) %>%
    rename(fecha = time, brecha = values) %>%
    arrange(desc(fecha))

gender_data$geo <- as.character(gender_data$geo)
gender_data$geo[ grepl("Union", gender_data$geo) ] <- "Unión Europea"
gender_data$geo[ grepl("Spain", gender_data$geo) ] <- "España"
```

Ahora, en la Figura 14, vamos a comprobar como se verían los datos del *dataset* «Brecha de género en la contratación», con el tratamiento previo y mediante la función «ggplot» de R:

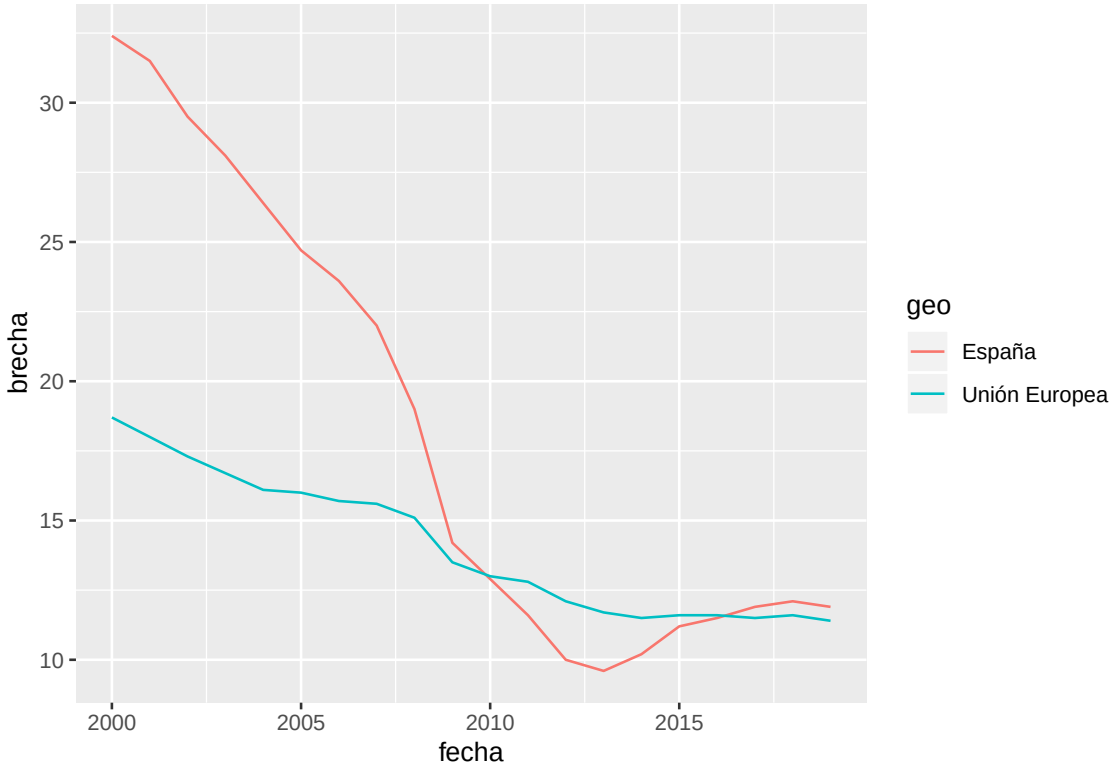

Figura 14: Brecha de género en la contratación (ggplot)

Sin embargo, para el informe hemos decidido utilizar la librería Shiny, que permite la incorporación de elementos interactivos, como un selector múltiple o un selector de rango. En este ejemplo, utilizamos el primero para seleccionar las áreas geográficas de estudio y el segundo para el rango de fechas.

Adicionalmente, incrustamos la función «ggplot» dentro de otra similar proveniente de la librería «plotly» (Sievert, 2020), que permite una presentación de los gráficos más clara y estilizada, dispone de una barra de ajustes de visualización y muestra los datos concretos al pasar el ratón por encima de ellos.

A continuación, en la Figura 15 mostramos los mismos datos que hemos visualizado antes, pero tal y como se muestran en la aplicación:

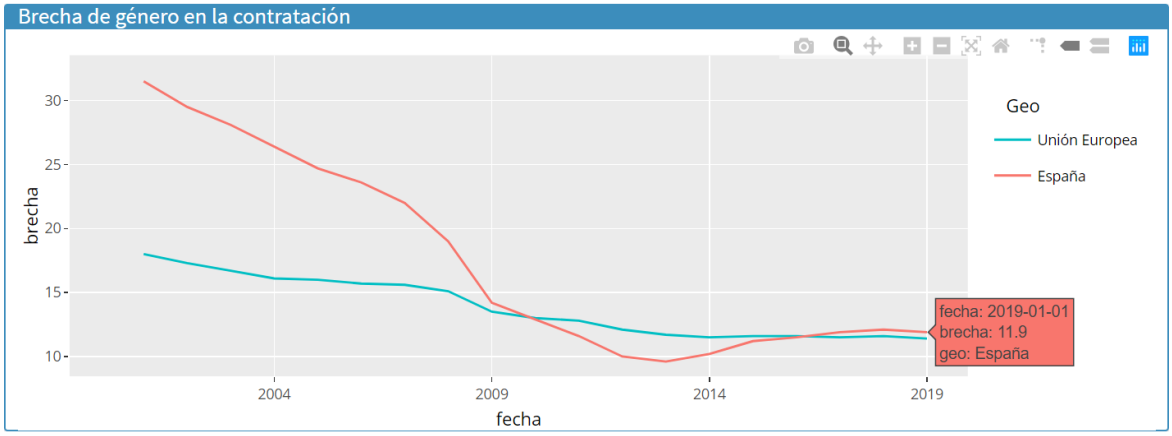

Figura 15: Brecha de género en la contratación (Shiny+Plotly)

Para finalizar este apartado, vamos a interpretar brevemente los resultados en base a los conocimientos adquiridos durante el grado:

Respecto a la UE, observamos que la pendiente ha permanecido negativa, llevando así a una disminución progresiva de la brecha. Esto se debe a la tendencia general de avanzar hacia una igualdad de género en lo laboral, promovida por numerosas directivas emitidas por la UE, diversos movimientos y múltiples organizaciones, como el Instituto Europeo de la Igualdad de Género (EIGE en inglés), que «trabaja para hacer realidad la igualdad de género en la UE y fuera de ella» (Unión Europea, 2020).

Sin embargo, el caso de España es sorprendentemente distinto. Desde el año 2000 hasta 2013 la brecha disminuye drásticamente. Observando los datos que proporciona el EIGE sobre los indicadores laborales en la igualdad de género en España, entre 2005 y 2013:

- Aumenta el ratio de empleo completo equivalente para las mujeres.
- El porcentaje de mujeres empleadas en la educación, salud y actividades de trabajo social crece 2.1 puntos porcentuales, mientras que para los hombres crece 1.1 puntos porcentuales.

Por otro lado, entre 2013 y 2019 la brecha ha aumentado hasta 2 puntos porcentuales. En este caso los datos del EIGE no son concluyentes, pero si estudiamos el informe de 2019 y separamos los indicadores según la estructura familiar, obtenemos la siguiente información:

- Existen importantes diferencias de género según el sector. Por ejemplo, en la «educación, salud y trabajos sociales» trabaja el 24% de las mujeres y solo el 7% de los hombres. Por contraste, en el sector de «ciencia, tecnología, ingeniería y matemáticas» trabaja el 5% de las mujeres respecto al 29% de los hombres.
- El porcentaje de población que trabaja a tiempo parcial es de un 24% para las mujeres y un 7% para los hombres.
- En parejas con hijos, el ratio de mujeres que trabajan a tiempo completo es un 57.4% mientras que el de los hombres es un 84%. Aquí se nota el efecto «ama de casa», es decir que aún es habitual en las relaciones de pareja que el hombre sea el que trabaja y la mujer la que cuida de los hijos.

En conclusión, a España aún le queda camino para alcanzar la igualdad de género, especialmente en el ámbito laboral. La adaptación de directrices y cambios legislativos representan pasos en la dirección adecuada, pero todavía falta que ocurran varios cambios culturales para poder llegar al equilibrio.

Destacamos que para realizar una interpretación completa lo correcto es desglosar el cálculo del índice en sus distintos componentes con el fin de averiguar cuales resultan ser los más influyentes en cada instante y contrastarlos con otras fuentes de información. Por ejemplo, en este caso el desglose podría realizarse por intervalos de edad, por sector laboral o por región geográfica.

4.2 Emisiones de gases de efecto invernadero

Los segundos datos provienen de la tabla llamada «*Greenhouse gas emissions*» («Emisiones de gases de efecto invernadero»), de código «sdg_13_10» y con la siguiente página con información.

El indicador mide las emisiones nacionales totales, integrando cada gas utilizando su potencial de calentamiento global. De nuevo, los metadatos nos permiten comprobar que la unidad de medida es un índice con base 100 en el año 1990 y los datos son anuales.

Igual que con el *dataset* anterior, empezamos extrayendo los datos dándoles nombres de variables inteligibles.

```
gas_data <- get_eurostat("sdg_13_10", type = "label")
```

También realizamos el ajuste de representación fiel de la UE, con la misma separación de 2020 y filtrando todo menos la UE y España.

```
gas_data <- filter(
  gas_data, time >= "2020-02-01"
              & geo == "European Union - 27 countries (from 2020)"
                      # Salida de Reino Unido
            | time < "2020-02-01"
              & geo == "European Union - 28 countries (2013-2020)"
            | geo %in% c("Spain"))
```

En este caso tenemos dos opciones para el indicador: en equivalente de CO2 y año base 1990 (valor de 100) o en toneladas por cápita. Para facilitar la comparación escogemos el primero, dejando solo un valor para la variable del indicador, por lo que lo eliminamos seleccionando las demás columnas.

```
gas_data <- filter(gas_data, grepl("CO2", indic_env))
gas_data <- select(gas_data, geo, time, values)
```

Finalmente resta realizar ajustes de estilo y traducciones al español. El código completo es el siguiente:

```
gas_data <- get_eurostat("sdg_13_10", type = "label") %>%
    filter(time >= "2020-02-01"
            & geo == "European Union - 27 countries (from 2020)"
                    # Salida de Reino Unido
           | time < "2020-02-01"
            & geo == "European Union - 28 countries (2013-2020)"
           | geo %in% c("Spain")) %>%
    filter(grepl("CO2", indic_env)) %>%
    select(geo, time, values) %>%
    rename(fecha = time, emisiones = values) %>%
    arrange(desc(fecha))

gas_data$geo <- as.character(gas_data$geo)
gas_data$geo[ grepl("Union", gas_data$geo) ] <- "Unión Europea"
gas_data$geo[ grepl("Spain", gas_data$geo) ] <- "España"
```

La Figura 16 muestra como se verían los datos del *dataset* «Emisiones de gases de efecto invernadero», tras aplicarles el tratamiento previo y mediante la función «ggplot» de R:

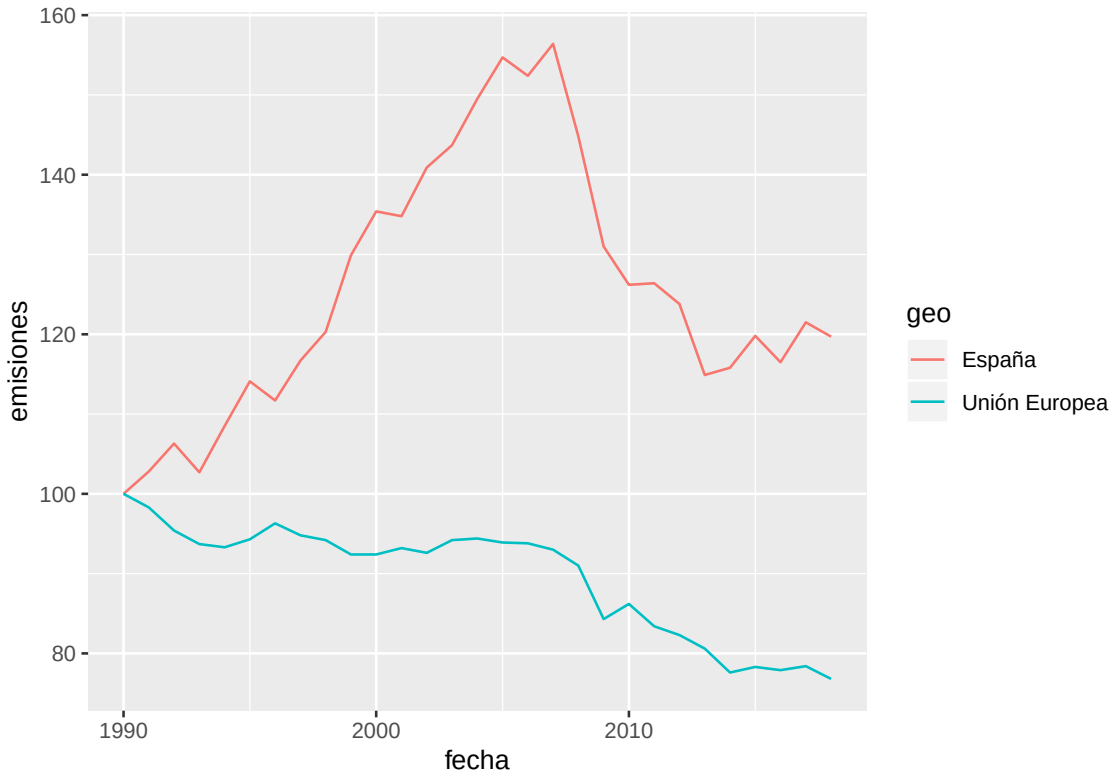

Figura 16: Emisiones de gases de efecto invernadero (ggplot)

Y de nuevo, vamos a compararlo con la forma en la que se ve utilizando Shiny y Plotly en la Figura 17:

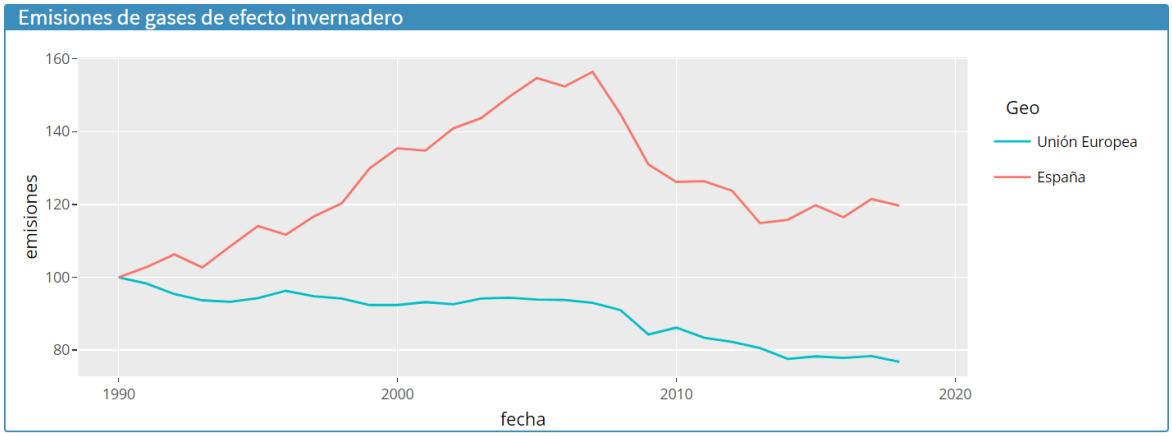

Figura 17: Emisiones de gases de efecto invernadero (Shiny+Plotly)

4.3 Resultado final

Ahora que hemos estudiado cada conjunto de datos individualmente, vamos a mostrar en la Figura 18 su apariencia en el informe interactivo. Aquí se pueden ver las gráficas anteriores acompañadas de *widgets* e información contextual:

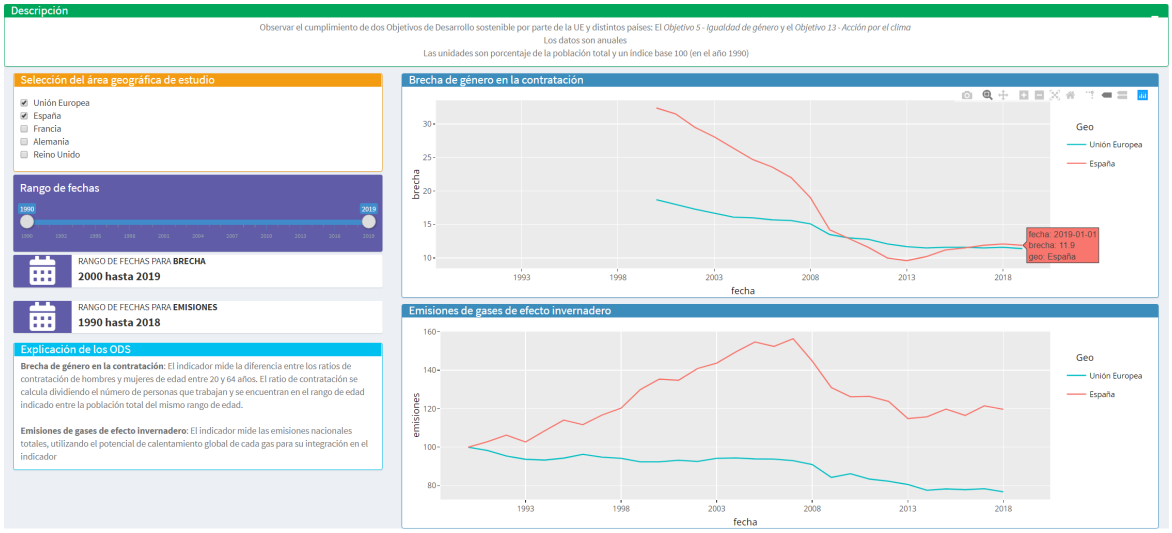

Figura 18: Pestaña ODS

Tras haber tratado con numerosos *datasets*, hemos solidificado nuestra creencia que mencionamos anteriormente: si se conocen los conjuntos de datos de los que se quiere extraer información actualizada, realizar informes estáticos es una opción ligeramente anticuada. A nuestro parecer es mucho más interesante presentar los informes de manera interactiva, permitiendo que el usuario disponga de toda la información y pueda focalizar en los puntos que le interesen. En nuestros informes, dichos puntos pueden ser, por ejemplo, un rango de fechas concreto o un subconjunto de variables de estudio.

4.4 Modelado ARIMA y predicción: PIB

Para finalizar este capítulo vamos a aplicar de forma práctica lo explicado en el apartado 2.5 Procedimiento de modelado ARIMA. Utilizaremos los datos del conjunto de datos «*GDP and main components (output, expenditure and income)*» visto en el subapartado Pestaña de 'PIB'. Hemos escogido este *dataset* por tener una periodicidad inferior al año (trimestral), ya que el modelado ARIMA requiere

que pueda existir una componente estacional.

Trabajaremos con valores del PIB, ya que se podría considerar como la variable de mayor interés del conjunto de datos. También trataremos con los datos sin aplicar la **eliminación de estacionalidad**, ya que la existencia de esta es un prerrequisito para el modelado ARIMA.

PIB de la Unión Europea

En primer lugar estudiaremos los valores de la UE, dado que es el área geográfica por excelencia en Eurostat.

Pasos 1 y 2: Representamos la serie temporal de datos en la Figura 19 para su posterior modelización:

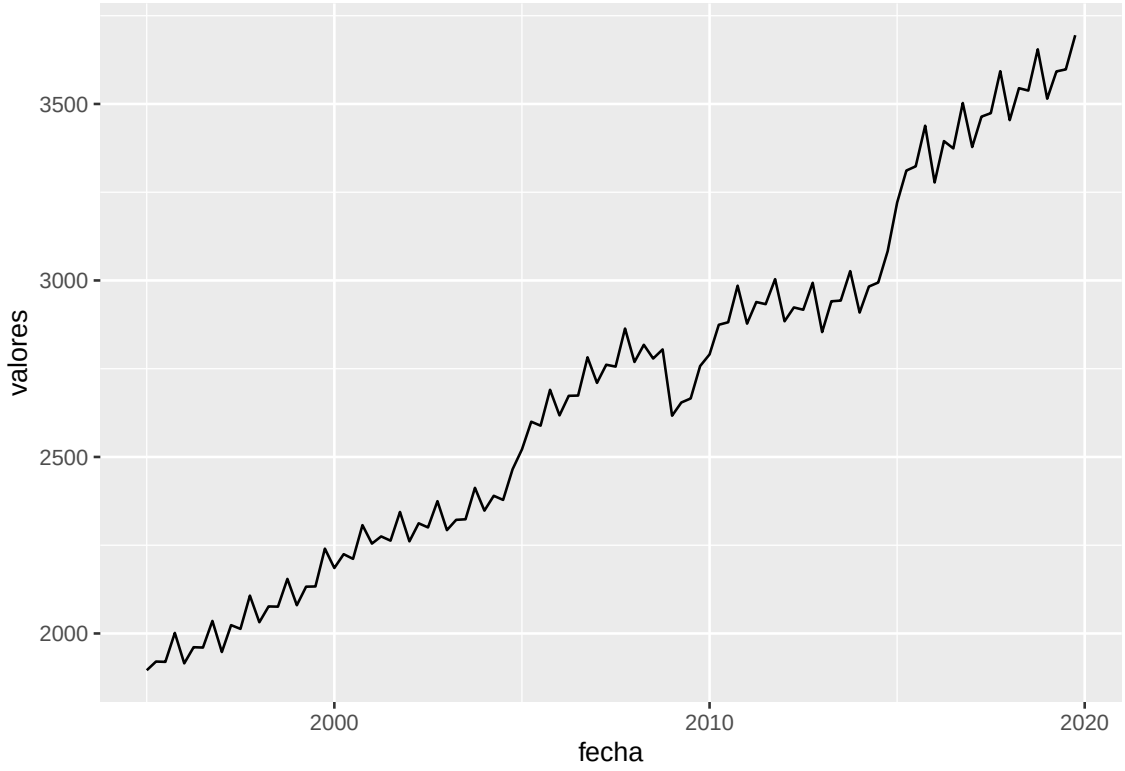

Figura 19: Evolución del PIB de la UE

No hay evidencias de que la varianza se modifique con el tiempo, así que no será necesario realizar una transformación Box-Cox.

Para obtener una predicción más acertada, en lugar de trabajar con el histórico de valores del PIB emplearemos la tasa de variación relativa. Se calcula como

$$\Delta Y_t = \frac{Y_t - Y_{t-1}}{Y_{t-1}} \tag{1}$$

Donde Y_t es el PIB registrado en el periodo t. Nuestro objetivo es predecir la tasa de variación relativa de cada periodo mediante el modelo ARIMA y utilizarlo para calcular el PIB correspondiente.

Los datos se encuentran almacenados en la variable «PIB_data», que procedemos a convertir en una serie temporal que reconozca R, tomando una única variable (la tasa de variación relativa), el año de partida y valores trimestrales.

```
Raw_data <- PIB_data %>% select(variacion)
ts_data = ts(Raw_data, start = c(1995,1), frequency = 4)
```

La Figura 20 muestra la apariencia de la serie temporal de la tasa de variación relativa del PIB:

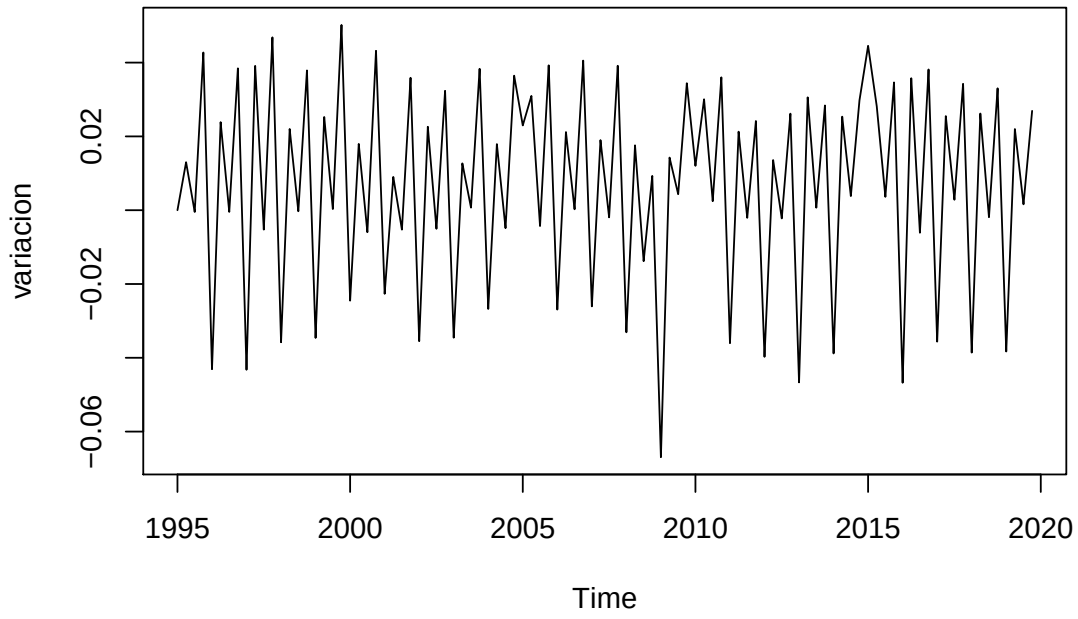

Figura 20: Evolución de la variación del PIB de la UE (serie temporal)

Paso 3: Obtenemos el modelo ARIMA mediante la función «auto.arima()». Establecemos el parámetro «approximation» como falso para que los criterios de información utilizados en la selección del modelo no sean aproximados, garantizando así la máxima verosimilitud en la estimación de los parámetros del modelo.

De forma similar, indicamos «stepwise» como falso para que el algoritmo busque entre todos los modelos posibles, en lugar de llevar a cabo una selección por pasos. Con estos parámetros conseguimos que el modelo resultante será lo más ajustado posible, a costa de un mayor tiempo de cómputo.

```
fit_ARIMA <- auto.arima(ts_data, trace = FALSE,
                        approximation = FALSE, stepwise = FALSE)
fit_ARIMA
```

```
## Series: ts_data
## ARIMA(1,0,0)(2,1,2)[4]
##
```

```
## Coefficients:
##          ar1     sar1     sar2     sma1     sma2
##       0.0820  -0.9460  -0.2421  -0.0730  -0.8299
## s.e.  0.1056   0.1258   0.1110   0.1427   0.1371
##
## sigma^2 estimated as 0.0001761:  log likelihood=275.49
## AIC=-538.99   AICc=-538.05   BIC=-523.6
```

El algoritmo prueba todas las combinaciones posibles de los parámetros p, d y q para seleccionar el mejor modelo, correspondiente al menor AIC. La traza completa del algoritmo se puede encontrar en el anexo Traza de la función auto.arima() para el PIB de la UE. En este caso, el modelo indicado es un ARIMA(1,0,0)(2,1,2). El [4] hace referencia al periodo de estacionalidad, puesto que como los datos son trimestrales, hay 4 en el año.

Paso 4: Realizamos diferentes tests para comprobar que los residuos son ruido blanco. En primer lugar mostramos en la Figura 21 las autocorrelaciones ACF de los residuos, que son mayoritariamente no significativas, aunque se evidencia autocorrelaciones aisladas:

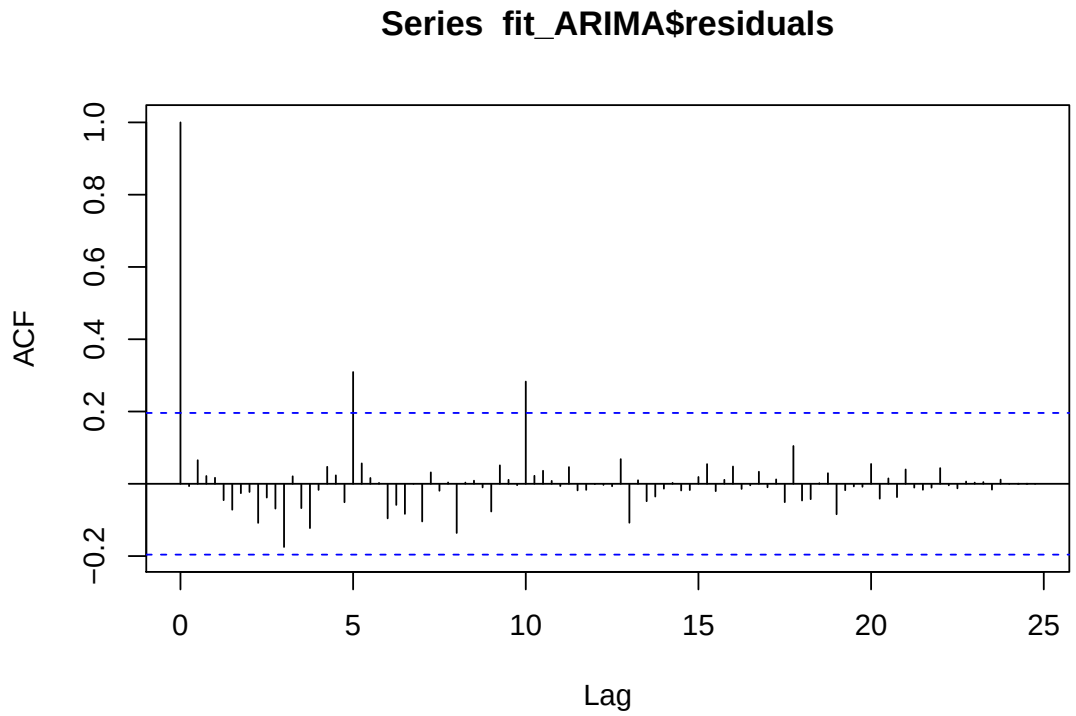

Figura 21: ACF de los residuos (UE)

En segundo lugar realizamos la prueba de Ljung-Box Q para verificar la independencia. Como podemos ver en la Figura 22, todos los valores de p están por encima de 0.05, lo que indica que no podemos rechazar la hipótesis nula de independencia de los residuos.

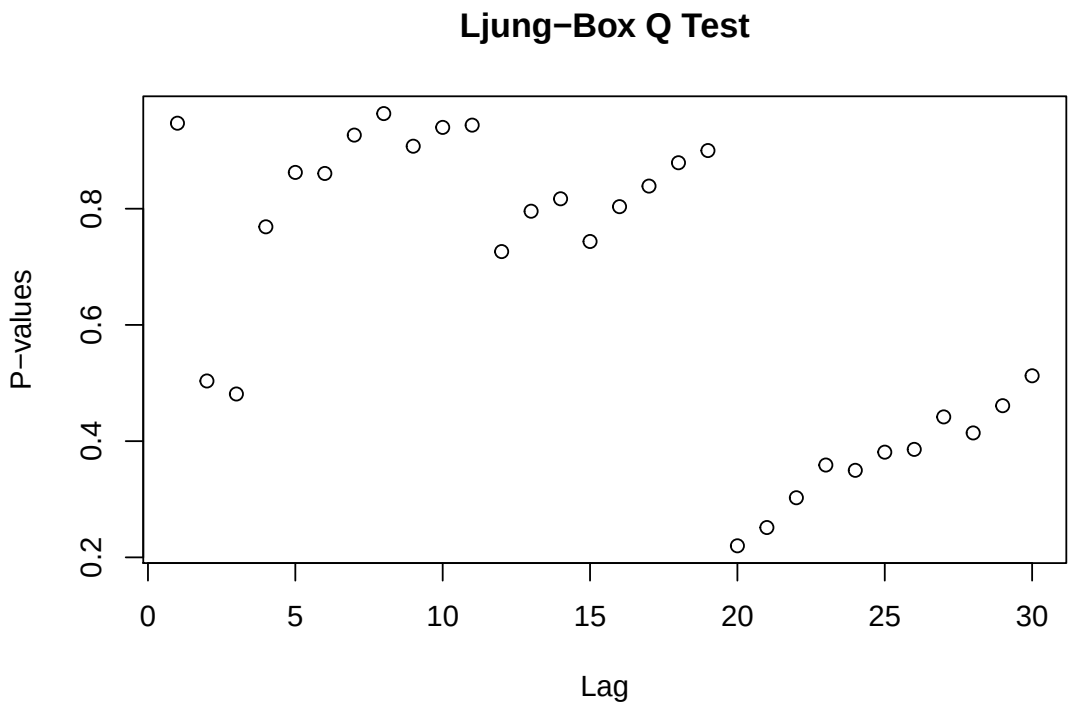

Figura 22: Prueba de Ljung-Box Q (UE)

En tercer lugar visualizamos en la Figura 23 la normalidad de los residuos en un gráfico Q-Q. Como los valores están en su mayoría alineados, podemos suponer normalidad aproximada. No obstante, se observan valores extremos que podrían depurarse.

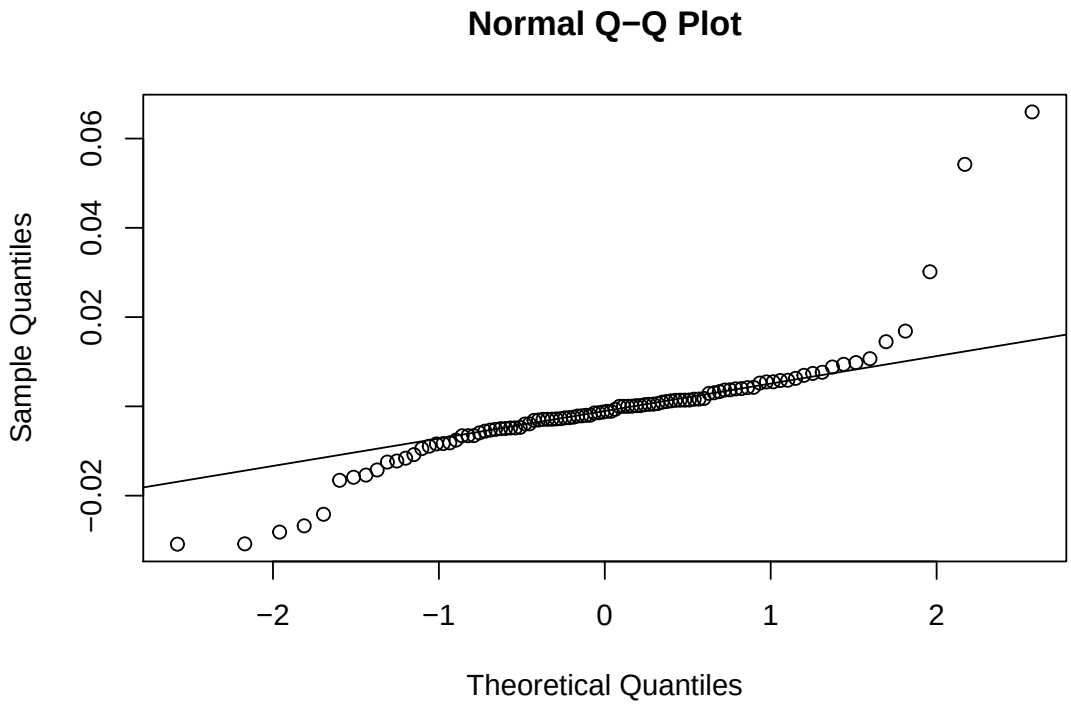

Figura 23: Gráfico Q-Q (UE)

Aunque los gráficos evidencian alguna deficiencia de cumplimiento de hipótesis, este modelo podría darnos una primera idea de la evolución de la tasa de variación relativa del PIB de la UE. Por lo tanto, podemos proceder a calcular la predicción.

Utilizando el modelo ARIMA y la función «forecast()», realizamos la predicción a 5 años (4 valores por año, 20 predicciones para 5 años) con un intervalo de confianza del 99.5%. El resultado es la Figura 24:

```
prediccion <- forecast(fit_ARIMA, h=20, level=c(99.5))
autoplot(prediccion)
```

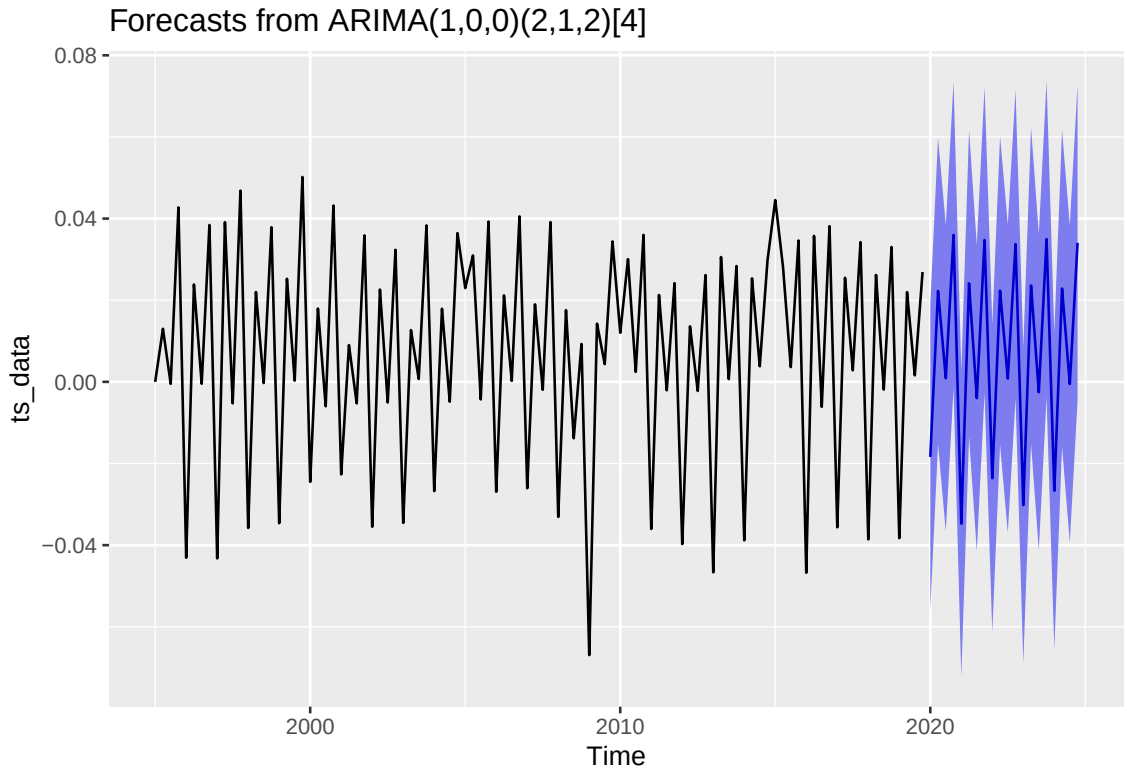

Figura 24: Predicción a 5 años de la variación del PIB de la UE

Para calcular los valores del PIB en base a la tasa de variación relativa predicha, utilizamos la fórmula anterior despejando Y_t:

$$Y_t = \Delta Y_t * Y_{t-1} + Y_{t-1} \tag{2}$$

Aplicamos este cálculo para predecir el PIB de la UE en los próximos 5 años y obtenemos así la Figura 25:

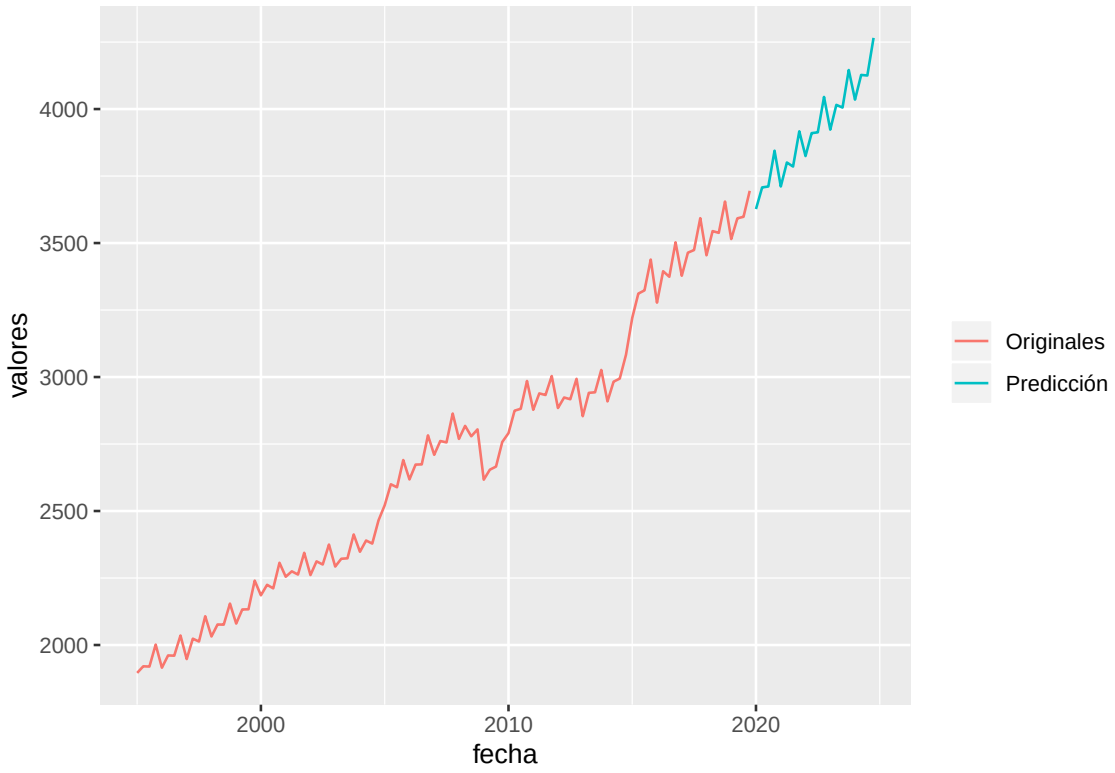

Figura 25: Predicción a 5 años del PIB de la UE

Según este modelo y como podemos ver en la Figura 25, la predicción del PIB en la UE para el 1 de octubre de 2024 es de 4265.635 miles de millones de euros.

Destacamos que la predicción tiene una serie de altibajos porque hemos utilizado el conjunto de datos tratado en el subapartado Pestaña de 'PIB' pero sin incluir la transformación que denominamos **eliminación de estacionalidad**. Por lo tanto, aunque el elemento más influyente sea la **tendencia** de la serie, la predicción refleja su estacionalidad mediante oscilaciones periódicas de periodo inferior a un año.

PIB de España

Para solidificar el modelado ARIMA y aportar una medida de comparación, vamos a realizar el mismo procedimiento con los datos del PIB de España.

Pasos 1 y 2: Representamos la serie temporal de datos en la Figura 26 para su posterior modelización:

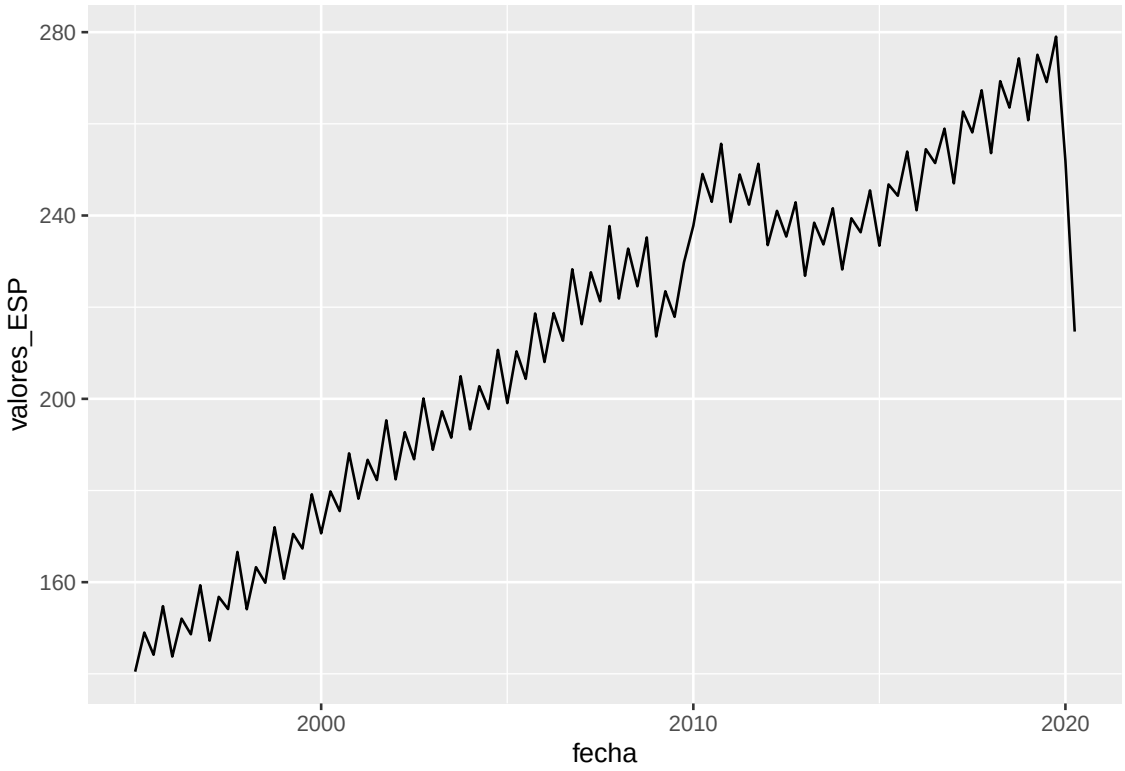

Figura 26: Evolución del PIB de España

En este caso los últimos datos corresponden al 1 de enero de 2020 y al 1 de abril de 2020. Podemos observar una caída en picado del PIB en estas dos últimas observaciones, lo que podría ser un indicador de que la varianza se modifica con el tiempo.

Sin embargo, este repentino descenso del PIB a comienzos de 2020 es real, como podemos comprobar en la página web del Instituto Nacional de Estadística. Por lo tanto, aunque altere de forma significativa la predicción, hemos decidido conservarlos. De esta forma podremos comparar la influencia de posibles datos anómalos con respecto a la predicción realizada anteriormente.

Igual que para la UE, vamos a predecir usando la variación en lugar del histórico del PIB. La Figura 27 muestra la apariencia de la serie temporal de la tasa de variación relativa del PIB:

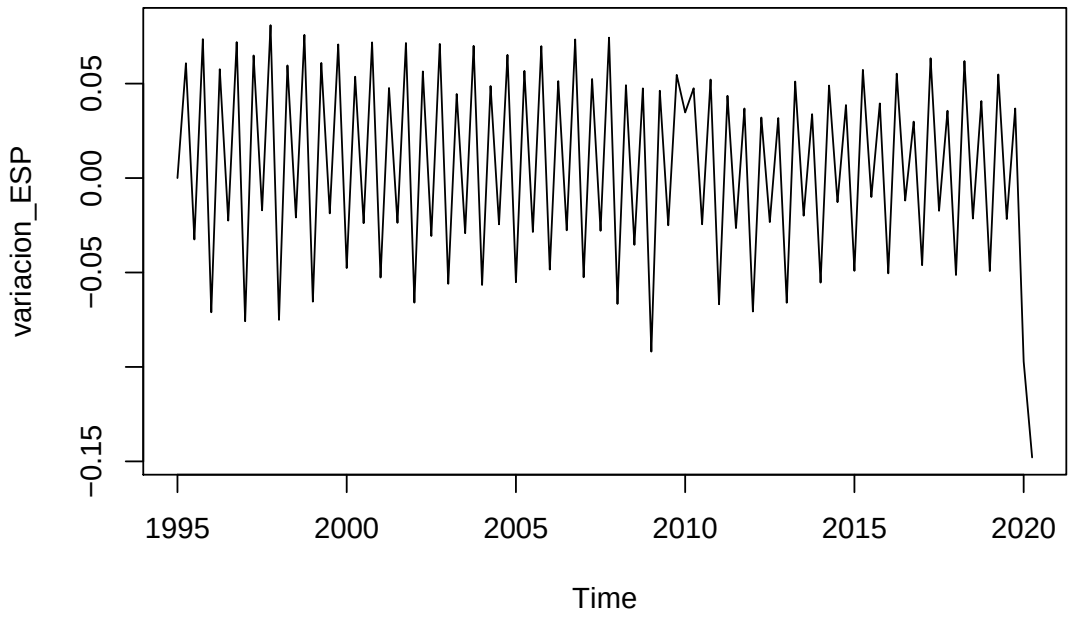

Figura 27: Evolución de la variación del PIB de España (serie temporal)

Paso 3: Obtenemos el modelo ARIMA mediante la función «auto.arima()», también estableciendo los parámetros «approximation» y «stepwise» como falsos para obtener el modelo más ajustado posible.

```
fit_ARIMA_ESP <- auto.arima(ts_data_ESP, trace = FALSE,
                            approximation = FALSE, stepwise = FALSE)
fit_ARIMA_ESP
```

```
## Series: ts_data_ESP
## ARIMA(1,0,0)(0,1,1)[4] with drift
##
## Coefficients:
##          ar1     sma1   drift
```

```
##       0.4602  -0.7650  -5e-04
## s.e.  0.1778   0.0958   4e-04
##
## sigma^2 estimated as 0.0006289:  log likelihood=221.83
## AIC=-435.66   AICc=-435.23   BIC=-425.32
```

De nuevo, la traza completa del algoritmo se puede encontrar en el anexo Traza de la función auto.arima() para el PIB de España. En este caso, el modelo indicado es un ARIMA(1,0,0)(0,1,1) con deriva, que es una constante que se incluye en el modelo sumando. Aquí tiene como valor -5e-4 , lo que es un valor muy cercano a cero. Igual que para la UE, el [4] hace referencia al periodo trimestral de estacionalidad.

Paso 4: Realizamos tests para comprobar que los residuos son ruido blanco. En primer lugar mostramos en la Figura 28 las autocorrelaciones ACF de los residuos, que son mayoritariamente no significativas, aunque se evidencia una autocorrelación aislada:

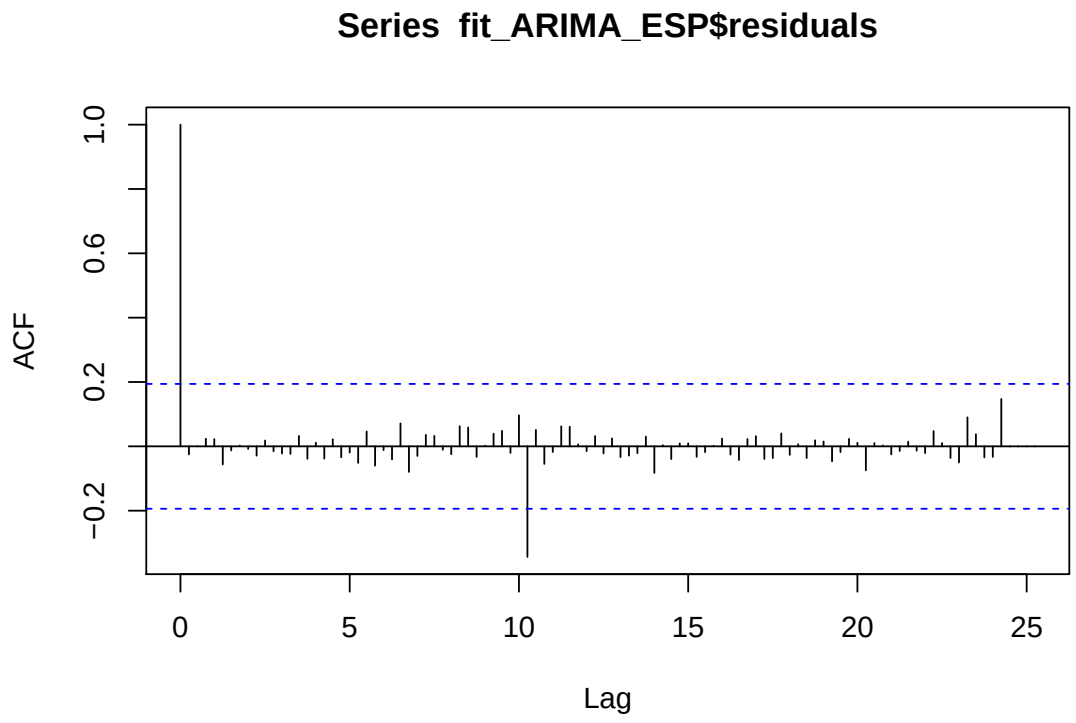

Figura 28: ACF de los residuos (España)

En segundo lugar realizamos la prueba de Ljung-Box Q para verificar la independencia. Como podemos ver en la Figura 29, todos los valores de p están muy por encima de 0.05, lo que indica que no podemos rechazar la hipótesis de independencia de los residuos.

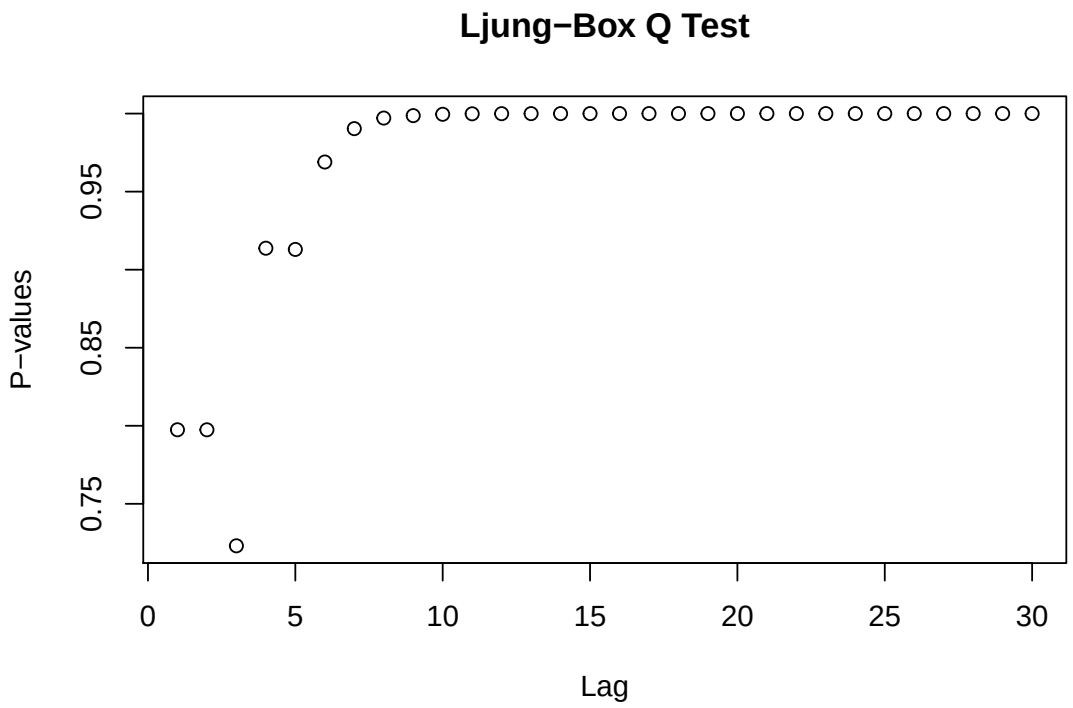

Figura 29: Prueba de Ljung-Box Q (España)

En tercer lugar visualizamos en la Figura 30 la normalidad de los residuos en un gráfico Q-Q. Como los valores están en su mayoría alineados, podemos suponer normalidad aproximada. No obstante, se observan algunos valores extremos que podrían depurarse.

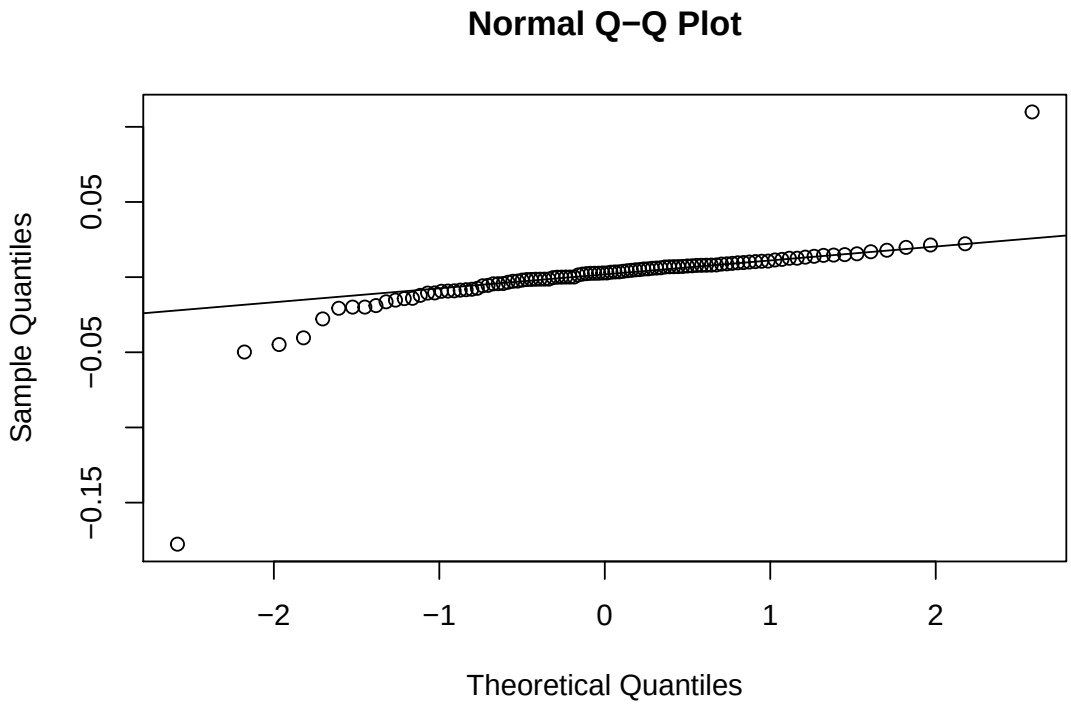

Figura 30: Gráfico Q-Q (España)

Aunque los gráficos evidencian alguna deficiencia de cumplimiento de hipótesis, este modelo podría darnos una primera idea de la evolución de la tasa de variación relativa del PIB de España. Por lo tanto, podemos proceder a calcular la predicción.

Utilizando el modelo ARIMA y la función «forecast()», realizamos la predicción a 5 años para que coincida con la de la UE (4 valores por año excluyendo los dos primeros, 18 predicciones en total) con un intervalo de confianza del 99.5%. El resultado es la Figura 31:

```
prediccion_ESP <- forecast(fit_ARIMA_ESP, h=18, level=c(99.5))
autoplot(prediccion_ESP)
```

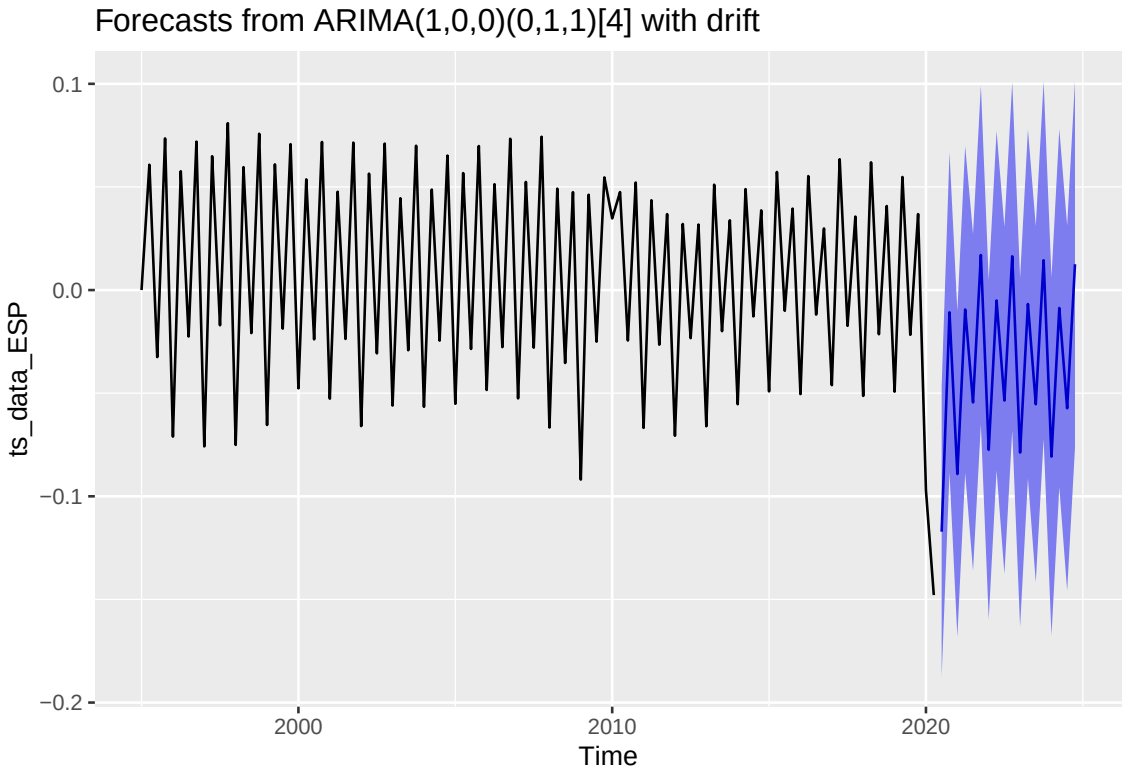

Figura 31: Predicción a 5 años de la tasa de variación del PIB de España

Volvemos a realizar el mismo cálculo para predecir el PIB de España en los próximos 5 años y obtenemos los datos representados en la Figura 32:

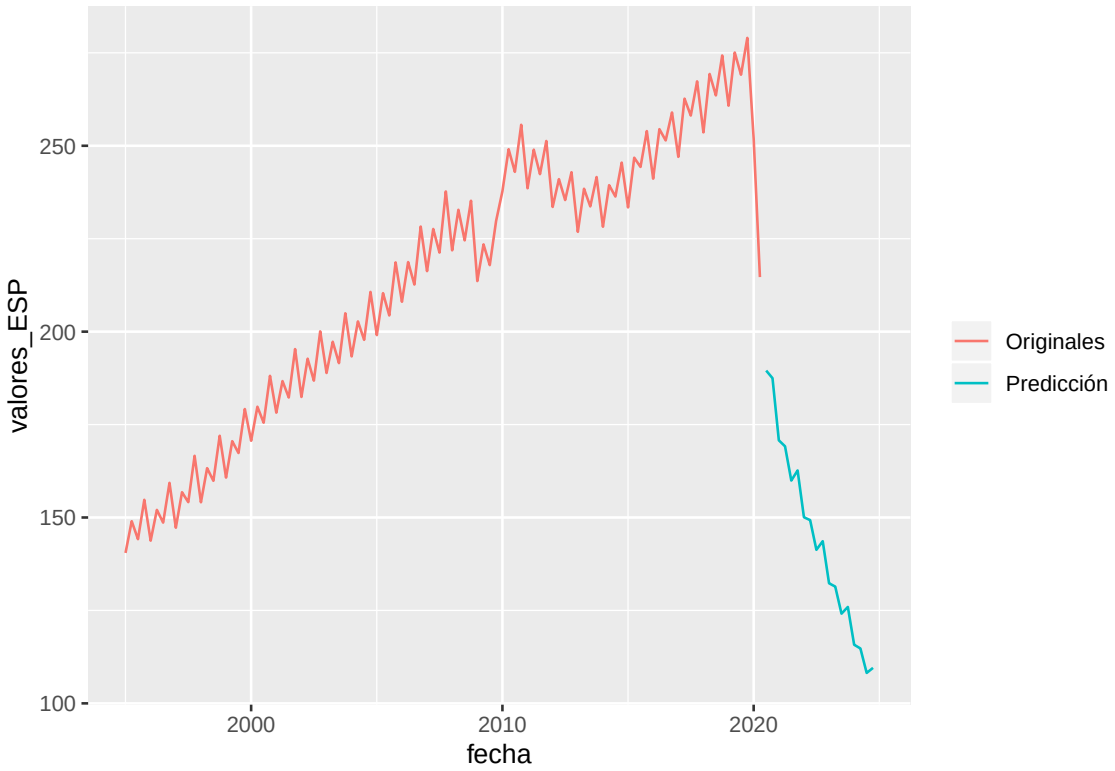

Figura 32: Predicción a 5 años del PIB de España

Según este modelo y como podemos ver en la Figura 32, la predicción del PIB de España para el 1 de octubre de 2024 es de 109.543 miles de millones de euros. También constatamos que el repentino descenso del PIB en los dos últimos datos ha tenido una importante influencia sobre la predicción.

Aunque parezca radical, hemos decidido conservar esta predicción porque se ajusta a la realidad que estamos viviendo a causa del COVID-19. Además, refleja el peso de los últimos datos y de la constancia de la varianza sobre el modelo ARIMA y por lo tanto sobre las predicciones.

Finalmente, para poner en perspectiva las diferencias de valores, evolución y predicciones de los PIB de la UE y de España, vamos a mostrar en la Figura 33 ambas series simultáneamente:

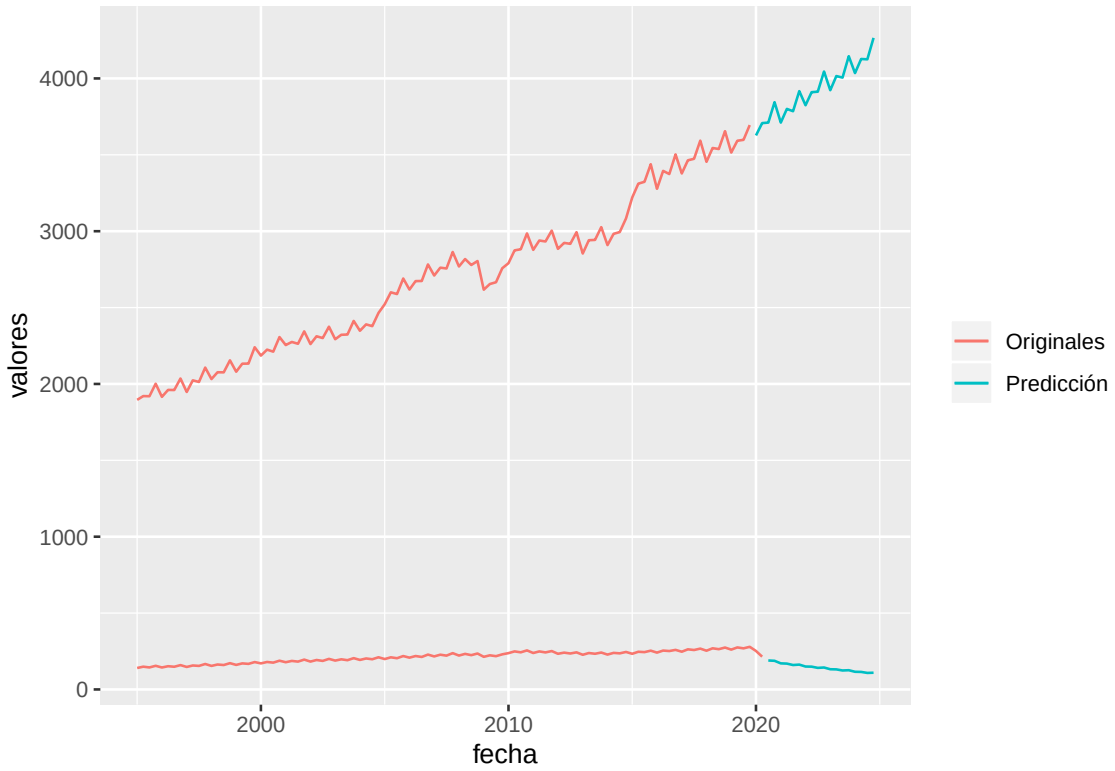

Figura 33: Predicción a 5 años del PIB de la UE y de España

Nos gustaría resaltar que en este apartado tan solo hemos mostrado un ejemplo de como se pueden utilizar herramientas como Eurostat y R para efectuar predicciones. Este mismo procedimiento se puede seguir para enriquecer más el proceso de *Business Analytics*, seleccionando múltiples variables concretas y comparando resultados entre ellas. Desgraciadamente, el alcance de nuestro trabajo no nos permite incluir modelizado y predicciones en el proceso de *Business Analytics*, así que nos hemos tenido que conformar con una demostración del potencial de las herramientas estudiadas.

5. Conclusiones

A lo largo del desarrollo de este proyecto hemos atravesado distintas fases, encontrando en cada una ideas antiguas y conceptos nuevos. Algunas fases nos han ocasionado frustración y muchas han concluido de forma satisfactoria.

En este capítulo haremos hincapié en los conocimientos adquiridos, las competencias obtenidas y los logros alcanzados. Pero también resaltaremos obstáculos que han surgido y cómo los hemos superado, llegando así al resultado final.

R y RStudio

En una primera instancia, hemos aprendido a manejar el lenguaje de programación R y el entorno de programación RStudio. Ambos han requerido de un periodo de adaptación, en el que hemos podido averiguar el extenso rango de funciones que proporcionan. En consecuencia, hemos seleccionado y profundizado en aquellas que nos serían de utilidad.

Nuestra guía ha sido el libro **R for Data Science** (Wickham y Grolemund, 2017), que nos ha introducido no solamente al lenguaje y entorno de programación, sino también a la disciplina de la ciencia de datos. Hemos aprendido las bases y la mentalidad con la que se debe enfocar el estudio de bases de datos y las hemos puesto en práctica mediante ejercicios y el propio trabajo.

De forma más específica, hemos aprendido a manejar archivos de los siguientes tipos y lenguajes de programación:

- R Markdown para la escritura de la memoria, empleando LateX para el estilo y la estructura.
- R junto con Shiny para el desarrollo de los informes, con pinceladas de HTML y CSS para el estilo.
- bibTex a través de Mendeley para las gestiones bibliográficas.

Eurostat

La segunda fase tenía como objetivo descubrir y explorar Eurostat para familiarizarse con su estructura y uso. La base de datos europea era prácticamente

desconocida para nosotros, pero hemos logrado dicho objetivo estudiando la página web e indagando en las funcionalidades del paquete «eurostat» para R.

Finalmente hemos conseguido seleccionar y extraer datos diversos y de actualidad, tratándolos para mejorar su visualización. Sin embargo, nos hemos tenido que conformar con tratar Eurostat solo para algunos ejemplos, ya que la extensión de este trabajo no es suficiente para abarcar una parte significativa de la inmensa base de datos europea.

Dicho esto, en el futuro seguramente vuelva a utilizar Eurostat o alguna otra base de datos internacional. Su fiabilidad y amplio rango de información las convierten en herramientas de suma importancia a la hora de tomar decisiones económico-empresariales. Este trabajo me ha permitido ganar soltura en el uso de bases de datos de gran utilidad para el *Business Analytics*, disciplina que con certeza aplicaré en el ámbito profesional y puede que incluso personal.

Ciencia de datos y Business Analytics

En el proceso de estudio de Eurostat hemos podido tratar datos reales, lo que conlleva numerosas implicaciones. Entre otras, hemos observado que los conjuntos de datos a menudo se optimizan para su almacenamiento, en lugar de para su estudio; y que generalmente en las bases de datos se almacenan datos de toda índole, así que para el estudio se debe escoger cuidadosamente aquellos que son de interés.

Hemos aprendido y aplicado técnicas de selección, tratamiento y visualización de datos, que constituyen el núcleo de la ciencia de datos y hemos interpretado los datos aplicando criterios aprendidos durante el grado, realizando así un ejercicio completo del *Business Analytics*.

Desarrollo de los informes

Finalmente llegamos a la fase de mayor envergadura: el desarrollo de los informes socioeconómicos interactivos. La labor consistía principalmente en programar, pero siempre hemos tenido en mente al usuario. Es por ello que hemos ido mejorando iterativamente la aplicación, para buscar un acceso más intuitivo e interactivo.

Hemos aprovechado la extensibilidad de R para utilizar paquetes que aporten funcionalidades más allá de las de las herramientas básicas, con el fin de mejorar

el producto final.

Sin embargo, también han surgido obstáculos, como el de la pestaña de 'Densidad de población'. Lo ideal habría sido poder visualizar los datos en el mapa de España, pero tras una exhaustiva investigación y numerosos intentos, hemos tenido que desistir. Decidimos priorizar la fluidez general de la aplicación y minimizar el tiempo de espera frente a una presentación un poco más atractiva.

Respecto al funcionamiento de los informes, al finalizar cada uno lo hemos sometido a pruebas extensivas hasta poder asegurar su robustez y claridad. El usuario objetivo posee conocimientos socioeconómicos básicos, pero hemos realizado pruebas con diversos usuarios. De esta forma garantizamos que prácticamente cualquier persona pueda utilizar la aplicación para interpretar los datos extraídos.

Crecimiento personal

Para cerrar el documento me gustaría hablar de lo que el desarrollo de este Trabajo de Fin de Grado ha supuesto para mí.

A pesar de estar en quinto de carrera, tenía la impresión de que carecía de práctica, lo que era una de las causas de mi indecisión respecto a mi futuro. Estaba deseoso de probar cosas nuevas, pero que tuviesen relación con mis estudios, a ser posible aprovechando ambas vertientes: informática y ADE.

Es entonces cuando descubrí una importante intersección entre ambas: la ciencia de datos, especialmente tratando datos socioeconómicos. Este trabajo me ha permitido profundizar en ella y ponerla en práctica utilizando una de las bases de datos más completas de la Unión Europea. Gracias a ello he podido refinar mis gustos, aceptando la ciencia de datos como una posible salida laboral.

Lo mismo podría decir de la Business Analytics y la importancia de los datos como ayuda a la toma de decisiones. Constituyen una filosofía y metodología que aplicaré en la medida de lo posible en mis futuros trabajos.

Por otra parte, la realización de este proyecto me ha aportado mucha independencia y soltura. Considero que mis tutores han realizado un gran trabajo aportándome recursos esenciales al comienzo y supervisando mi progreso, pero finalmente el peso de la investigación y desarrollo recae sobre mi.

He aprendido a manejar herramientas útiles a las que probablemente recurra de nuevo en un futuro, pero también he mejorado muchas de mis *soft skills*. Definitivamente, gracias a este trabajo podré afrontar proyectos futuros con mayor

seguridad y entusiasmo.

Referencias bibliográficas

Acito, F., y Khatri, V. (2014). *Business analytics: Why now and what next?* Elsevier Ltd. https://doi.org/10.1016/j.bushor.2014.06.001

Arnold, J. B. (2020). *R for Data Science : Exercise Solutions* (pp. 1-220). Recuperado de https://jrnold.github.io/r4ds-exercise-solutions/index.html

Brockwell, P. J., y Davis, R. A. (2002). *Introduction to time series and forecasting* (3.ª ed., Vol. 2). Springer. https://doi.org/10.1007/978-3-319-29854-2

Campos, F. (2018). *Análisis de series temporales en R. ARIMA*. Recuperado de https://www.diegocalvo.es/analisis-de-series-temporales-en-r-arima/ el 12 de agosto de 2020

Chang, W., y Borges Ribeiro, B. (2018). *shinydashboard: Create Dashboards with 'Shiny'*. Recuperado de https://CRAN.R-project.org/package=shinydashboard

Chang, W., Cheng, J., Allaire, J., Xie, Y., y McPherson, J. (2020). *shiny: Web Application Framework for R*. Recuperado de https://CRAN.R-project.org/package=shiny

Chatterjee, S. (2018). *Time Series Analysis Using ARIMA Model In R*. Recuperado de https://datascienceplus.com/time-series-analysis-using-arima-model-in-r/ el 12 de agosto de 2020

Chirivella, V. (2018). *Apuntes de Econometría. Número 12059-C*. Universidad Politécnica de Valencia.

Codecademy. (2020). *Codecademy*. Recuperado de https://www.codecademy.com/ el 16 de junio de 2020

Date, S. (2020). *The Akaike Information Criterion*. Recuperado de https://towardsdatascience.com/the-akaike-information-criterion-c20c8fd832f2 el 12 de agosto de 2020

DeSignificado.com. (2020). *Significado de Navaja de Ockham*. Recuperado de https://designificado.com/navaja-de-ockham/ el 12 de agosto de 2020

European Commission. (2018). *Eurostat - History of NUTS*. Recuperado de https://ec.europa.eu/eurostat/web/nuts/history el 16 de junio de 2020

European Commission. (2020a). *Eurostat - Frequently asked questions*. Re-

cuperado de https://ec.europa.eu/eurostat/help/faq el 20 de junio de 2020

European Commission. (2020b). *Eurostat - Publishing statistics after the United Kingdom left the EU*. Recuperado de https://ec.europa.eu/eurostat/web/products-eurostat-news/-/WDN-20200201-1 el 22 de junio de 2020

European Commission. (2020c). *Job Vacancies*. Recuperado de https://ec.europa.eu/eurostat/web/labour-market/job-vacancies el 18 de junio de 2020

European Union. (2020). *Countries | European Union*. Recuperado de https://europa.eu/european-union/about-eu/countries_en#tab-0-1 el 17 de junio de 2020

Eurostat. (2018). *Eurostat - La llave de acceso a las estadísticas europeas*. Recuperado de https://ec.europa.eu/eurostat/documents/4031688/8932118/KS-02-17-839-ES-N.pdf/04b4200a-eba6-477f-8690-b47a308e7e20#:$/sim$:text=La%20misi%7B/'%7Bo%7D%7Dn%20de%20Eurostat%20es%20proporcionar%20estad%7B/'%7Bi%7D%7Dsticas%20de%20alta%20calidad%20para%20Europa.&text=Eurostat%20proporciona%20estad%7B/'%7Bi%7D%7Dsticas%20a%20nivel%20europ

Eurostat. (2020). *Seasonal adjustment - Eurostat*. Recuperado de https://ec.europa.eu/eurostat/web/research-methodology/seasonal-adjustment el 30 de julio de 2020

Font Awesome. (2020). *Icons | Font Awesome*. Recuperado de https://fontawesome.com/icons?d=gallery el 17 de junio de 2020

Granger, C. W. J., Newbold, P., y Shell, K. (2014). *Forecasting Economic Time Series* (2.ª ed.; K. Shell, Ed.). Elsevier Science. Recuperado de https://books.google.es/books?id=oDWjBQAAQBAJ

Hyndman, R., Athanasopoulos, G., Bergmeir, C., Caceres, G., Chhay, L., O'Hara-Wild, M., … Yasmeen, F. (2020). *forecast: Forecasting functions for time series and linear models*. Recuperado de http://pkg.robjhyndman.com/forecast

Hyndman, R. J., y Athanasopoulos, G. (2018). *Forecasting: principles and practice - ARIMA modelling in R* (2.ª ed.). OTexts: Melbourne, Australia. Recuperado de https://otexts.com/fpp2/arima-r.html

Hyndman, R., y Khandakar, Y. (2008). Automatic Time Series Forecasting: The forecast Package for R. *Journal of Statistical Software, Articles*, *27*(3), 1-22. https://doi.org/10.18637/jss.v027.i03

IBM Knowledge Center. (2020). *Funciones de autocorrelación y autocorrelación parcial*. Recuperado de https://www.ibm.com/support/knowledgecenter/

es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/dt/timeseries_acf_pacf.html el 14 de agosto de 2020

Instituto Vasco de Estadística. (2020a). *Definicion de Gasto en consumo final*. Recuperado de https://www.eustat.eus/documentos/opt_0/tema_478/elem_3563/definicion.html el 16 de junio de 2020

Instituto Vasco de Estadística. (2020b). *Definición de Exportaciones de bienes y servicios*. Recuperado de https://www.eustat.eus/documentos/opt_0/tema_155/elem_2524/definicion.html el 16 de junio de 2020

Instituto Vasco de Estadística. (2020c). *Definición de Importaciones de bienes y servicios*. Recuperado de https://www.eustat.eus/documentos/opt_0/tema_155/elem_2526/definicion.html el 16 de junio de 2020

Kutner, M. H. (2005). *Applied Linear Statistical Models*. McGraw-Hill Irwin. Recuperado de https://books.google.es/books?id=0xqCAAAACAAJ

Lahti, L., Huovari, J., Kainu, M., y Biecek, P. (2017). eurostat R package. Recuperado de https://journal.r-project.org/archive/2017/RJ-2017-019/index.html

López, J. F. (2019). *Varianza - Qué es, definición y significado | Economipedia*. Recuperado de https://economipedia.com/definiciones/varianza.html el 12 de agosto de 2020

Naciones Unidas. (2020). *Objetivos y metas de desarrollo sostenible*. Recuperado de https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/ el 1 de julio de 2020

Oracle. (2020). *¿Qué es la ciencia de datos?* Recuperado de https://www.oracle.com/es/data-science/what-is-data-science.html el 22 de junio de 2020

Overleaf. (2019). *Overleaf, Online LaTeX editor*. Recuperado de https://www.overleaf.com/ el 16 de junio de 2020

Plotly. (2020). *Getting Started with Plotly for R*. Recuperado de https://plotly.com/r/getting-started/ el 17 de junio de 2020

R Core Team. (2020). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. Recuperado de https://www.R-project.org/

RStudio. (2020). *Shiny Widget Gallery*. Recuperado de https://shiny.rstudio.com/gallery/widget-gallery.html el 16 de junio de 2020

RStudio Team. (2019). *RStudio: Integrated Development Environment for R*. Boston, MA: RStudio, Inc. Recuperado de http://www.rstudio.com/

Sánchez Galán, J. (2020). *Formación bruta de capital fijo - Qué es, definición y concepto*. Recuperado de https://economipedia.com/definiciones/formacion-bruta-de-capital-fijo.html el 16 de junio de 2020

Seddon, P. B., Constantinidis, D., Tamm, T., y Dod, H. (2017). How does business analytics contribute to business value? *Information Systems Journal*, *27*(3), 237-269. https://doi.org/10.1111/isj.12101

Sevilla, A. (2012). Producto interior bruto (PIB) - Qué es, definición y significado. *Economipedia*. Recuperado de https://economipedia.com/definiciones/producto-interior-bruto-pib.html

Sievert, C. (2020). *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman; Hall/CRC. Recuperado de https://plotly-r.com

Unión Europea. (2020). *European Institute for Gender Equality (EIGE)*. Recuperado de https://europa.eu/european-union/about-eu/agencies/eige_es el 11 de agosto de 2020

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. Recuperado de https://ggplot2.tidyverse.org

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Wickham, H., y Grolemund, G. (2017). *R for Data Science*. https://doi.org/10.18637/jss.v077.b01

Wikipedia. (2020a). *Eurostat - Wikipedia*. Recuperado de https://en.wikipedia.org/wiki/Eurostat el 16 de junio de 2020

Wikipedia. (2020b). *Nomenclatura de las Unidades Territoriales Estadísticas*. Recuperado de https://es.wikipedia.org/wiki/Nomenclatura_de_las_Unidades_Territoriales_Estad%7B/'%7Bi%7D%7Dsticas el 24 de junio de 2020

Wikipedia. (2020c). *NUTS de España*. Recuperado de https://es.wikipedia.org/wiki/NUTS_de_Espa%7B/~%7Bn%7D%7Da el 24 de junio de 2020

Yihui, X. (2017). *Shiny - How to use DataTables in a Shiny App*. Recuperado de https://shiny.rstudio.com/articles/datatables.html el 27 de junio de 2020

Yihui, X. (2020). *DataTables Options*. Recuperado de https://rstudio.github.io/DT/options.html el 27 de junio de 2020

Anexos

Traza de la función auto.arima() para el PIB de la UE

```
##
##  ARIMA(0,0,0)(0,1,0)[4]                    : -477.8464
##  ARIMA(0,0,0)(0,1,0)[4] with drift         : -475.8088
##  ARIMA(0,0,0)(0,1,1)[4]                    : Inf
##  ARIMA(0,0,0)(0,1,1)[4] with drift         : Inf
##  ARIMA(0,0,0)(0,1,2)[4]                    : Inf
##  ARIMA(0,0,0)(0,1,2)[4] with drift         : Inf
##  ARIMA(0,0,0)(1,1,0)[4]                    : -502.2797
##  ARIMA(0,0,0)(1,1,0)[4] with drift         : -500.2149
##  ARIMA(0,0,0)(1,1,1)[4]                    : Inf
##  ARIMA(0,0,0)(1,1,1)[4] with drift         : Inf
##  ARIMA(0,0,0)(1,1,2)[4]                    : -537.6673
##  ARIMA(0,0,0)(1,1,2)[4] with drift         : -535.5841
##  ARIMA(0,0,0)(2,1,0)[4]                    : -512.6214
##  ARIMA(0,0,0)(2,1,0)[4] with drift         : -510.5365
##  ARIMA(0,0,0)(2,1,1)[4]                    : -536.9374
##  ARIMA(0,0,0)(2,1,1)[4] with drift         : -534.8505
##  ARIMA(0,0,0)(2,1,2)[4]                    : Inf
##  ARIMA(0,0,0)(2,1,2)[4] with drift         : Inf
##  ARIMA(0,0,1)(0,1,0)[4]                    : -477.0332
##  ARIMA(0,0,1)(0,1,0)[4] with drift         : -474.9526
##  ARIMA(0,0,1)(0,1,1)[4]                    : Inf
##  ARIMA(0,0,1)(0,1,1)[4] with drift         : Inf
##  ARIMA(0,0,1)(0,1,2)[4]                    : Inf
##  ARIMA(0,0,1)(0,1,2)[4] with drift         : Inf
##  ARIMA(0,0,1)(1,1,0)[4]                    : -501.5415
##  ARIMA(0,0,1)(1,1,0)[4] with drift         : -499.4337
##  ARIMA(0,0,1)(1,1,1)[4]                    : Inf
##  ARIMA(0,0,1)(1,1,1)[4] with drift         : Inf
##  ARIMA(0,0,1)(1,1,2)[4]                    : -536.1066
##  ARIMA(0,0,1)(1,1,2)[4] with drift         : -533.9692
##  ARIMA(0,0,1)(2,1,0)[4]                    : -511.9154
##  ARIMA(0,0,1)(2,1,0)[4] with drift         : -509.7855
##  ARIMA(0,0,1)(2,1,1)[4]                    : -535.4006
##  ARIMA(0,0,1)(2,1,1)[4] with drift         : -533.2613
##  ARIMA(0,0,1)(2,1,2)[4]                    : -537.9651
##  ARIMA(0,0,1)(2,1,2)[4] with drift         : -535.7752
```

```
##  ARIMA(0,0,2)(0,1,0)[4]                    : -488.9572
##  ARIMA(0,0,2)(0,1,0)[4] with drift         : -486.8038
##  ARIMA(0,0,2)(0,1,1)[4]                    : Inf
##  ARIMA(0,0,2)(0,1,1)[4] with drift         : Inf
##  ARIMA(0,0,2)(0,1,2)[4]                    : -535.214
##  ARIMA(0,0,2)(0,1,2)[4] with drift         : Inf
##  ARIMA(0,0,2)(1,1,0)[4]                    : -500.8454
##  ARIMA(0,0,2)(1,1,0)[4] with drift         : -498.6837
##  ARIMA(0,0,2)(1,1,1)[4]                    : Inf
##  ARIMA(0,0,2)(1,1,1)[4] with drift         : Inf
##  ARIMA(0,0,2)(1,1,2)[4]                    : -534.4387
##  ARIMA(0,0,2)(1,1,2)[4] with drift         : -532.1044
##  ARIMA(0,0,2)(2,1,0)[4]                    : -510.7049
##  ARIMA(0,0,2)(2,1,0)[4] with drift         : -508.5179
##  ARIMA(0,0,2)(2,1,1)[4]                    : -533.6437
##  ARIMA(0,0,2)(2,1,1)[4] with drift         : -531.4413
##  ARIMA(0,0,3)(0,1,0)[4]                    : -489.6833
##  ARIMA(0,0,3)(0,1,0)[4] with drift         : -487.5063
##  ARIMA(0,0,3)(0,1,1)[4]                    : Inf
##  ARIMA(0,0,3)(0,1,1)[4] with drift         : Inf
##  ARIMA(0,0,3)(0,1,2)[4]                    : -533.1075
##  ARIMA(0,0,3)(0,1,2)[4] with drift         : -530.918
##  ARIMA(0,0,3)(1,1,0)[4]                    : -499.3137
##  ARIMA(0,0,3)(1,1,0)[4] with drift         : -497.1018
##  ARIMA(0,0,3)(1,1,1)[4]                    : Inf
##  ARIMA(0,0,3)(1,1,1)[4] with drift         : Inf
##  ARIMA(0,0,3)(2,1,0)[4]                    : -509.7092
##  ARIMA(0,0,3)(2,1,0)[4] with drift         : -507.4699
##  ARIMA(1,0,0)(0,1,0)[4]                    : -477.2071
##  ARIMA(1,0,0)(0,1,0)[4] with drift         : -475.1266
##  ARIMA(1,0,0)(0,1,1)[4]                    : Inf
##  ARIMA(1,0,0)(0,1,1)[4] with drift         : Inf
##  ARIMA(1,0,0)(0,1,2)[4]                    : Inf
##  ARIMA(1,0,0)(0,1,2)[4] with drift         : Inf
##  ARIMA(1,0,0)(1,1,0)[4]                    : -501.8197
##  ARIMA(1,0,0)(1,1,0)[4] with drift         : -499.712
##  ARIMA(1,0,0)(1,1,1)[4]                    : Inf
##  ARIMA(1,0,0)(1,1,1)[4] with drift         : Inf
##  ARIMA(1,0,0)(1,1,2)[4]                    : -536.2061
##  ARIMA(1,0,0)(1,1,2)[4] with drift         : -534.0685
##  ARIMA(1,0,0)(2,1,0)[4]                    : -512.2077
##  ARIMA(1,0,0)(2,1,0)[4] with drift         : -510.0772
##  ARIMA(1,0,0)(2,1,1)[4]                    : -535.501
```

```
##  ARIMA(1,0,0)(2,1,1)[4] with drift         : -533.36
##  ARIMA(1,0,0)(2,1,2)[4]                    : -538.0457
##  ARIMA(1,0,0)(2,1,2)[4] with drift         : -535.8547
##  ARIMA(1,0,1)(0,1,0)[4]                    : Inf
##  ARIMA(1,0,1)(0,1,0)[4] with drift         : Inf
##  ARIMA(1,0,1)(0,1,1)[4]                    : Inf
##  ARIMA(1,0,1)(0,1,1)[4] with drift         : Inf
##  ARIMA(1,0,1)(0,1,2)[4]                    : -535.1988
##  ARIMA(1,0,1)(0,1,2)[4] with drift         : -533.0655
##  ARIMA(1,0,1)(1,1,0)[4]                    : Inf
##  ARIMA(1,0,1)(1,1,0)[4] with drift         : Inf
##  ARIMA(1,0,1)(1,1,1)[4]                    : Inf
##  ARIMA(1,0,1)(1,1,1)[4] with drift         : Inf
##  ARIMA(1,0,1)(1,1,2)[4]                    : -534.2973
##  ARIMA(1,0,1)(1,1,2)[4] with drift         : -532.0417
##  ARIMA(1,0,1)(2,1,0)[4]                    : -510.7168
##  ARIMA(1,0,1)(2,1,0)[4] with drift         : -508.5316
##  ARIMA(1,0,1)(2,1,1)[4]                    : -533.5686
##  ARIMA(1,0,1)(2,1,1)[4] with drift         : -531.3677
##  ARIMA(1,0,2)(0,1,0)[4]                    : -486.7826
##  ARIMA(1,0,2)(0,1,0)[4] with drift         : -484.5807
##  ARIMA(1,0,2)(0,1,1)[4]                    : Inf
##  ARIMA(1,0,2)(0,1,1)[4] with drift         : Inf
##  ARIMA(1,0,2)(0,1,2)[4]                    : -533.094
##  ARIMA(1,0,2)(0,1,2)[4] with drift         : -530.9034
##  ARIMA(1,0,2)(1,1,0)[4]                    : -498.813
##  ARIMA(1,0,2)(1,1,0)[4] with drift         : -496.6007
##  ARIMA(1,0,2)(1,1,1)[4]                    : Inf
##  ARIMA(1,0,2)(1,1,1)[4] with drift         : Inf
##  ARIMA(1,0,2)(2,1,0)[4]                    : -508.9435
##  ARIMA(1,0,2)(2,1,0)[4] with drift         : -506.7028
##  ARIMA(1,0,3)(0,1,0)[4]                    : Inf
##  ARIMA(1,0,3)(0,1,0)[4] with drift         : Inf
##  ARIMA(1,0,3)(0,1,1)[4]                    : Inf
##  ARIMA(1,0,3)(0,1,1)[4] with drift         : Inf
##  ARIMA(1,0,3)(1,1,0)[4]                    : Inf
##  ARIMA(1,0,3)(1,1,0)[4] with drift         : Inf
##  ARIMA(2,0,0)(0,1,0)[4]                    : -475.4262
##  ARIMA(2,0,0)(0,1,0)[4] with drift         : -473.2975
##  ARIMA(2,0,0)(0,1,1)[4]                    : Inf
##  ARIMA(2,0,0)(0,1,1)[4] with drift         : Inf
##  ARIMA(2,0,0)(0,1,2)[4]                    : -535.3128
##  ARIMA(2,0,0)(0,1,2)[4] with drift         : -533.1773
```

```
##  ARIMA(2,0,0)(1,1,0)[4]                    : -500.495
##  ARIMA(2,0,0)(1,1,0)[4] with drift         : -498.3348
##  ARIMA(2,0,0)(1,1,1)[4]                    : Inf
##  ARIMA(2,0,0)(1,1,1)[4] with drift         : Inf
##  ARIMA(2,0,0)(1,1,2)[4]                    : Inf
##  ARIMA(2,0,0)(1,1,2)[4] with drift         : Inf
##  ARIMA(2,0,0)(2,1,0)[4]                    : -510.9254
##  ARIMA(2,0,0)(2,1,0)[4] with drift         : -508.7382
##  ARIMA(2,0,0)(2,1,1)[4]                    : -533.703
##  ARIMA(2,0,0)(2,1,1)[4] with drift         : -531.4989
##  ARIMA(2,0,1)(0,1,0)[4]                    : -473.2641
##  ARIMA(2,0,1)(0,1,0)[4] with drift         : -471.087
##  ARIMA(2,0,1)(0,1,1)[4]                    : Inf
##  ARIMA(2,0,1)(0,1,1)[4] with drift         : Inf
##  ARIMA(2,0,1)(0,1,2)[4]                    : -533.0747
##  ARIMA(2,0,1)(0,1,2)[4] with drift         : -530.8847
##  ARIMA(2,0,1)(1,1,0)[4]                    : -498.2679
##  ARIMA(2,0,1)(1,1,0)[4] with drift         : -496.0576
##  ARIMA(2,0,1)(1,1,1)[4]                    : Inf
##  ARIMA(2,0,1)(1,1,1)[4] with drift         : Inf
##  ARIMA(2,0,1)(2,1,0)[4]                    : -508.7547
##  ARIMA(2,0,1)(2,1,0)[4] with drift         : -506.5149
##  ARIMA(2,0,2)(0,1,0)[4]                    : Inf
##  ARIMA(2,0,2)(0,1,0)[4] with drift         : Inf
##  ARIMA(2,0,2)(0,1,1)[4]                    : Inf
##  ARIMA(2,0,2)(0,1,1)[4] with drift         : Inf
##  ARIMA(2,0,2)(1,1,0)[4]                    : Inf
##  ARIMA(2,0,2)(1,1,0)[4] with drift         : Inf
##  ARIMA(2,0,3)(0,1,0)[4]                    : Inf
##  ARIMA(2,0,3)(0,1,0)[4] with drift         : Inf
##  ARIMA(3,0,0)(0,1,0)[4]                    : -473.5344
##  ARIMA(3,0,0)(0,1,0)[4] with drift         : -471.358
##  ARIMA(3,0,0)(0,1,1)[4]                    : Inf
##  ARIMA(3,0,0)(0,1,1)[4] with drift         : Inf
##  ARIMA(3,0,0)(0,1,2)[4]                    : -533.0854
##  ARIMA(3,0,0)(0,1,2)[4] with drift         : -530.8951
##  ARIMA(3,0,0)(1,1,0)[4]                    : -498.2679
##  ARIMA(3,0,0)(1,1,0)[4] with drift         : -496.0577
##  ARIMA(3,0,0)(1,1,1)[4]                    : Inf
##  ARIMA(3,0,0)(1,1,1)[4] with drift         : Inf
##  ARIMA(3,0,0)(2,1,0)[4]                    : -508.987
##  ARIMA(3,0,0)(2,1,0)[4] with drift         : -506.7469
##  ARIMA(3,0,1)(0,1,0)[4]                    : Inf
```

```
##  ARIMA(3,0,1)(0,1,0)[4] with drift         : Inf
##  ARIMA(3,0,1)(0,1,1)[4]                    : Inf
##  ARIMA(3,0,1)(0,1,1)[4] with drift         : Inf
##  ARIMA(3,0,1)(1,1,0)[4]                    : Inf
##  ARIMA(3,0,1)(1,1,0)[4] with drift         : Inf
##  ARIMA(3,0,2)(0,1,0)[4]                    : Inf
##  ARIMA(3,0,2)(0,1,0)[4] with drift         : Inf
##
##
##
##  Best model: ARIMA(1,0,0)(2,1,2)[4]


## Series: ts_data
## ARIMA(1,0,0)(2,1,2)[4]
##
## Coefficients:
##          ar1     sar1     sar2     sma1     sma2
##       0.0820  -0.9460  -0.2421  -0.0730  -0.8299
## s.e.  0.1056   0.1258   0.1110   0.1427   0.1371
##
## sigma^2 estimated as 0.0001761:  log likelihood=275.49
## AIC=-538.99   AICc=-538.05   BIC=-523.6
```

Traza de la función auto.arima() para el PIB de España

```
##
##  ARIMA(0,0,0)(0,1,0)[4]                    : -416.8293
##  ARIMA(0,0,0)(0,1,0)[4] with drift         : -416.1311
##  ARIMA(0,0,0)(0,1,1)[4]                    : -429.5453
##  ARIMA(0,0,0)(0,1,1)[4] with drift         : -431.0082
##  ARIMA(0,0,0)(0,1,2)[4]                    : -427.4288
##  ARIMA(0,0,0)(0,1,2)[4] with drift         : -428.9145
##  ARIMA(0,0,0)(1,1,0)[4]                    : -422.2397
##  ARIMA(0,0,0)(1,1,0)[4] with drift         : -421.6995
##  ARIMA(0,0,0)(1,1,1)[4]                    : -427.4265
##  ARIMA(0,0,0)(1,1,1)[4] with drift         : -428.9052
##  ARIMA(0,0,0)(1,1,2)[4]                    : -425.2699
##  ARIMA(0,0,0)(1,1,2)[4] with drift         : -426.703
##  ARIMA(0,0,0)(2,1,0)[4]                    : -422.6715
##  ARIMA(0,0,0)(2,1,0)[4] with drift         : -422.2607
##  ARIMA(0,0,0)(2,1,1)[4]                    : -425.4371
##  ARIMA(0,0,0)(2,1,1)[4] with drift         : -426.7754
##  ARIMA(0,0,0)(2,1,2)[4]                    : -424.1252
##  ARIMA(0,0,0)(2,1,2)[4] with drift         : -424.7275
##  ARIMA(0,0,1)(0,1,0)[4]                    : -420.2424
##  ARIMA(0,0,1)(0,1,0)[4] with drift         : -419.4039
##  ARIMA(0,0,1)(0,1,1)[4]                    : -433.7325
##  ARIMA(0,0,1)(0,1,1)[4] with drift         : -434.0381
##  ARIMA(0,0,1)(0,1,2)[4]                    : -431.5759
##  ARIMA(0,0,1)(0,1,2)[4] with drift         : -431.8745
##  ARIMA(0,0,1)(1,1,0)[4]                    : -426.6066
##  ARIMA(0,0,1)(1,1,0)[4] with drift         : -425.9164
##  ARIMA(0,0,1)(1,1,1)[4]                    : -431.5726
##  ARIMA(0,0,1)(1,1,1)[4] with drift         : -431.8667
##  ARIMA(0,0,1)(1,1,2)[4]                    : -429.3705
##  ARIMA(0,0,1)(1,1,2)[4] with drift         : -429.618
##  ARIMA(0,0,1)(2,1,0)[4]                    : -427.5242
##  ARIMA(0,0,1)(2,1,0)[4] with drift         : -426.9321
##  ARIMA(0,0,1)(2,1,1)[4]                    : -429.5435
##  ARIMA(0,0,1)(2,1,1)[4] with drift         : -429.7193
##  ARIMA(0,0,1)(2,1,2)[4]                    : -428.8248
##  ARIMA(0,0,1)(2,1,2)[4] with drift         : -427.833
##  ARIMA(0,0,2)(0,1,0)[4]                    : -418.3076
##  ARIMA(0,0,2)(0,1,0)[4] with drift         : -417.4104
##  ARIMA(0,0,2)(0,1,1)[4]                    : -432.3003
```

```
##  ARIMA(0,0,2)(0,1,1)[4] with drift         : -432.6044
##  ARIMA(0,0,2)(0,1,2)[4]                    : -430.0786
##  ARIMA(0,0,2)(0,1,2)[4] with drift         : -430.3613
##  ARIMA(0,0,2)(1,1,0)[4]                    : -424.6261
##  ARIMA(0,0,2)(1,1,0)[4] with drift         : -423.8806
##  ARIMA(0,0,2)(1,1,1)[4]                    : -430.0785
##  ARIMA(0,0,2)(1,1,1)[4] with drift         : -430.3582
##  ARIMA(0,0,2)(1,1,2)[4]                    : -427.8083
##  ARIMA(0,0,2)(1,1,2)[4] with drift         : -428.0472
##  ARIMA(0,0,2)(2,1,0)[4]                    : -425.5497
##  ARIMA(0,0,2)(2,1,0)[4] with drift         : -424.8947
##  ARIMA(0,0,2)(2,1,1)[4]                    : -427.9616
##  ARIMA(0,0,2)(2,1,1)[4] with drift         : -428.1191
##  ARIMA(0,0,3)(0,1,0)[4]                    : -429.4481
##  ARIMA(0,0,3)(0,1,0)[4] with drift         : -428.5268
##  ARIMA(0,0,3)(0,1,1)[4]                    : -430.76
##  ARIMA(0,0,3)(0,1,1)[4] with drift         : -430.6714
##  ARIMA(0,0,3)(0,1,2)[4]                    : -429.0606
##  ARIMA(0,0,3)(0,1,2)[4] with drift         : -428.855
##  ARIMA(0,0,3)(1,1,0)[4]                    : -427.7145
##  ARIMA(0,0,3)(1,1,0)[4] with drift         : -426.7923
##  ARIMA(0,0,3)(1,1,1)[4]                    : -428.9539
##  ARIMA(0,0,3)(1,1,1)[4] with drift         : -428.7948
##  ARIMA(0,0,3)(2,1,0)[4]                    : -426.7665
##  ARIMA(0,0,3)(2,1,0)[4] with drift         : -425.8996
##  ARIMA(1,0,0)(0,1,0)[4]                    : -420.5483
##  ARIMA(1,0,0)(0,1,0)[4] with drift         : -419.6973
##  ARIMA(1,0,0)(0,1,1)[4]                    : -435.1165
##  ARIMA(1,0,0)(0,1,1)[4] with drift         : -435.2301
##  ARIMA(1,0,0)(0,1,2)[4]                    : -432.9499
##  ARIMA(1,0,0)(0,1,2)[4] with drift         : -433.0538
##  ARIMA(1,0,0)(1,1,0)[4]                    : -427.3796
##  ARIMA(1,0,0)(1,1,0)[4] with drift         : -426.6467
##  ARIMA(1,0,0)(1,1,1)[4]                    : -432.9486
##  ARIMA(1,0,0)(1,1,1)[4] with drift         : -433.0488
##  ARIMA(1,0,0)(1,1,2)[4]                    : -430.7289
##  ARIMA(1,0,0)(1,1,2)[4] with drift         : -430.7908
##  ARIMA(1,0,0)(2,1,0)[4]                    : -428.3759
##  ARIMA(1,0,0)(2,1,0)[4] with drift         : -427.7158
##  ARIMA(1,0,0)(2,1,1)[4]                    : -430.8755
##  ARIMA(1,0,0)(2,1,1)[4] with drift         : -430.8611
##  ARIMA(1,0,0)(2,1,2)[4]                    : -429.6645
##  ARIMA(1,0,0)(2,1,2)[4] with drift         : -428.6948
```

```
##  ARIMA(1,0,1)(0,1,0)[4]                    : -418.4626
##  ARIMA(1,0,1)(0,1,0)[4] with drift         : -417.5628
##  ARIMA(1,0,1)(0,1,1)[4]                    : -433.1191
##  ARIMA(1,0,1)(0,1,1)[4] with drift         : -433.1167
##  ARIMA(1,0,1)(0,1,2)[4]                    : -430.9044
##  ARIMA(1,0,1)(0,1,2)[4] with drift         : -430.8937
##  ARIMA(1,0,1)(1,1,0)[4]                    : -425.264
##  ARIMA(1,0,1)(1,1,0)[4] with drift         : -424.4745
##  ARIMA(1,0,1)(1,1,1)[4]                    : -430.9033
##  ARIMA(1,0,1)(1,1,1)[4] with drift         : -430.889
##  ARIMA(1,0,1)(1,1,2)[4]                    : -428.6397
##  ARIMA(1,0,1)(1,1,2)[4] with drift         : -428.5798
##  ARIMA(1,0,1)(2,1,0)[4]                    : -426.1955
##  ARIMA(1,0,1)(2,1,0)[4] with drift         : -425.4733
##  ARIMA(1,0,1)(2,1,1)[4]                    : -428.7623
##  ARIMA(1,0,1)(2,1,1)[4] with drift         : -428.6353
##  ARIMA(1,0,2)(0,1,0)[4]                    : -416.4026
##  ARIMA(1,0,2)(0,1,0)[4] with drift         : -415.4588
##  ARIMA(1,0,2)(0,1,1)[4]                    : -430.9203
##  ARIMA(1,0,2)(0,1,1)[4] with drift         : -430.8474
##  ARIMA(1,0,2)(0,1,2)[4]                    : -428.6519
##  ARIMA(1,0,2)(0,1,2)[4] with drift         : -428.5725
##  ARIMA(1,0,2)(1,1,0)[4]                    : -423.0741
##  ARIMA(1,0,2)(1,1,0)[4] with drift         : -422.2317
##  ARIMA(1,0,2)(1,1,1)[4]                    : -428.6515
##  ARIMA(1,0,2)(1,1,1)[4] with drift         : -428.5677
##  ARIMA(1,0,2)(2,1,0)[4]                    : -424.0168
##  ARIMA(1,0,2)(2,1,0)[4] with drift         : -423.2312
##  ARIMA(1,0,3)(0,1,0)[4]                    : -429.5732
##  ARIMA(1,0,3)(0,1,0)[4] with drift         : -428.6165
##  ARIMA(1,0,3)(0,1,1)[4]                    : -428.875
##  ARIMA(1,0,3)(0,1,1)[4] with drift         : -428.6892
##  ARIMA(1,0,3)(1,1,0)[4]                    : -427.3563
##  ARIMA(1,0,3)(1,1,0)[4] with drift         : -426.3301
##  ARIMA(2,0,0)(0,1,0)[4]                    : -418.4638
##  ARIMA(2,0,0)(0,1,0)[4] with drift         : -417.5642
##  ARIMA(2,0,0)(0,1,1)[4]                    : -433.0779
##  ARIMA(2,0,0)(0,1,1)[4] with drift         : -433.1045
##  ARIMA(2,0,0)(0,1,2)[4]                    : -430.8666
##  ARIMA(2,0,0)(0,1,2)[4] with drift         : -430.8866
##  ARIMA(2,0,0)(1,1,0)[4]                    : -425.2476
##  ARIMA(2,0,0)(1,1,0)[4] with drift         : -424.4612
##  ARIMA(2,0,0)(1,1,1)[4]                    : -430.865
```

```
##  ARIMA(2,0,0)(1,1,1)[4] with drift         : -430.8816
##  ARIMA(2,0,0)(1,1,2)[4]                    : -428.6051
##  ARIMA(2,0,0)(1,1,2)[4] with drift         : -428.5728
##  ARIMA(2,0,0)(2,1,0)[4]                    : -426.1809
##  ARIMA(2,0,0)(2,1,0)[4] with drift         : -425.4638
##  ARIMA(2,0,0)(2,1,1)[4]                    : -428.7266
##  ARIMA(2,0,0)(2,1,1)[4] with drift         : -428.6276
##  ARIMA(2,0,1)(0,1,0)[4]                    : -416.2906
##  ARIMA(2,0,1)(0,1,0)[4] with drift         : Inf
##  ARIMA(2,0,1)(0,1,1)[4]                    : -430.9066
##  ARIMA(2,0,1)(0,1,1)[4] with drift         : -430.8462
##  ARIMA(2,0,1)(0,1,2)[4]                    : -428.6412
##  ARIMA(2,0,1)(0,1,2)[4] with drift         : -428.5724
##  ARIMA(2,0,1)(1,1,0)[4]                    : -423.0476
##  ARIMA(2,0,1)(1,1,0)[4] with drift         : -422.2085
##  ARIMA(2,0,1)(1,1,1)[4]                    : -428.6406
##  ARIMA(2,0,1)(1,1,1)[4] with drift         : -428.567
##  ARIMA(2,0,1)(2,1,0)[4]                    : -423.9431
##  ARIMA(2,0,1)(2,1,0)[4] with drift         : -423.1658
##  ARIMA(2,0,2)(0,1,0)[4]                    : -418.434
##  ARIMA(2,0,2)(0,1,0)[4] with drift         : -417.5875
##  ARIMA(2,0,2)(0,1,1)[4]                    : -428.6855
##  ARIMA(2,0,2)(0,1,1)[4] with drift         : Inf
##  ARIMA(2,0,2)(1,1,0)[4]                    : -421.7061
##  ARIMA(2,0,2)(1,1,0)[4] with drift         : Inf
##  ARIMA(2,0,3)(0,1,0)[4]                    : Inf
##  ARIMA(2,0,3)(0,1,0)[4] with drift         : Inf
##  ARIMA(3,0,0)(0,1,0)[4]                    : -416.3372
##  ARIMA(3,0,0)(0,1,0)[4] with drift         : -415.3841
##  ARIMA(3,0,0)(0,1,1)[4]                    : -431.0068
##  ARIMA(3,0,0)(0,1,1)[4] with drift         : -430.877
##  ARIMA(3,0,0)(0,1,2)[4]                    : -428.738
##  ARIMA(3,0,0)(0,1,2)[4] with drift         : -428.5937
##  ARIMA(3,0,0)(1,1,0)[4]                    : -423.2625
##  ARIMA(3,0,0)(1,1,0)[4] with drift         : -422.4033
##  ARIMA(3,0,0)(1,1,1)[4]                    : -428.7377
##  ARIMA(3,0,0)(1,1,1)[4] with drift         : -428.5898
##  ARIMA(3,0,0)(2,1,0)[4]                    : -424.2077
##  ARIMA(3,0,0)(2,1,0)[4] with drift         : -423.4007
##  ARIMA(3,0,1)(0,1,0)[4]                    : -421.9994
##  ARIMA(3,0,1)(0,1,0)[4] with drift         : -421.0213
##  ARIMA(3,0,1)(0,1,1)[4]                    : Inf
##  ARIMA(3,0,1)(0,1,1)[4] with drift         : -428.5627
```

```
##  ARIMA(3,0,1)(1,1,0)[4]                    : -424.0904
##  ARIMA(3,0,1)(1,1,0)[4] with drift         : -423.1797
##  ARIMA(3,0,2)(0,1,0)[4]                    : -417.2928
##  ARIMA(3,0,2)(0,1,0)[4] with drift         : Inf
##
##
##
##  Best model: ARIMA(1,0,0)(0,1,1)[4] with drift


## Series: ts_data_ESP
## ARIMA(1,0,0)(0,1,1)[4] with drift
##
## Coefficients:
##          ar1     sma1   drift
##       0.4602  -0.7650  -5e-04
## s.e.  0.1778   0.0958   4e-04
##
## sigma^2 estimated as 0.0006289:  log likelihood=221.83
## AIC=-435.66   AICc=-435.23   BIC=-425.32
```

Código de la aplicación (app.R)

```
#
# This is a Shiny web application. You can run the application by clicking
# the 'Run App' button above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#

list.of.packages <- c("tidyverse", "eurostat", "shinydashboard", "plotly", "lubridate", "DT")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)

library(tidyverse)
library(eurostat)
library(shinydashboard)
library(plotly)
library(lubridate)
library(DT)


# Define UI for application that draws a histogram
ui <- dashboardPage(skin = "blue",
  dashboardHeader(title = span("Eurostat: Informes socioeconómicos interactivos", style = "color: yellow"), titleWidth = 453),
  dashboardSidebar(
    sidebarMenu(
      # sidebarSearchForm(textId = "search_text", buttonId = "search_button", label = "Buscar..."),
      menuItem("Introducción", tabName = "Introducción", icon = icon("info")),
      menuItem("Visión general", tabName = "Vision_general", icon = icon("eye")),
      menuItem("PIB", tabName = "PIB", icon = icon("euro")),
      menuItem("Consumidores", tabName = "Consumidores", icon = icon("address-card")),
      menuItem("Empleo", tabName = "Empleo", icon = icon("adjust")),
      menuItem("Densidad de población", tabName = "Densidad_población", icon = icon("globe")),
      menuItem("ODS", tabName = "ODS", icon = icon("hand-holding-heart"))
      # https://fontawesome.com/icons?d=gallery
    )
  ),
  dashboardBody(
    tags$head(tags$style(HTML('.info-box {min-height: 50px;} .info-box-icon {height: 50px; line-height: 50px;}
                              .info-box-content {padding-top: 0px; padding-bottom: 0px;}
                              .box-header {height: 20px; padding-top: 0px; padding-bottom: 0px;}
                              .box-body {padding-top: 0px; padding-bottom: 0px;}
                              .box {margin-top: 5px; margin-bottom: 5px;}
                              .checkbox {margin-top: 5px; margin-bottom: 5px;}
                              .main-header {max-height: 20px; margin-bottom: 20px; padding-bottom: 20px;}
                              '))),

    tabItems(

      # Pestaña de Introducción-----------------------------------------------------------------------------------------------
      tabItem(tabName = "Introducción",
              fluidRow(
                box(title = "Introducción", status = "primary", solidHeader = TRUE, align = "center", width = "adjust",
                    helpText(h1( HTML( "<B>TFG: Utilización de la base de datos Eurostat. Elaboración de informes automáticos con R</B> <br>"))),
                    helpText(h2( HTML( "Autor: <em>Fernando Reboyras Soleto</em>"))),
                    helpText(h2( HTML("Tutores: <em>Francisco Javier Ribal Sanchis <br> y Ana María Debón Aucejo</em>"))),
                    helpText(h3( HTML( "Grado en Administración y Dirección de Empresas en la <br> Universidad Politécnica de Valencia
                                        <br> Curso 2019-2020")))
                ),
                box(title = "Descripción", status = "success", solidHeader = TRUE, align = "left", width = "adjust",
                    helpText( HTML( "Esta aplicación ha sido desarrollada con el fin de aprender y mostrar algunos de los datos de
                    la base de datos europea Eurostat. <br>
                    Se han utilizado elementos de las siguientes librerías: <br><br>
                    <ul>
                    <li><B>'eurostat'</B> para la selección y extracción de los datos </li><br>
                    <li><B>'tidyverse'</B> para el tratamiento y presentación de los datos </li><br>
                    <li><B>'shiny'</B> y <B>'shinydashboard'</B> para la presentación e interactividad de la interfaz</li><br>
                    <li><B>'plotly'</B> y <B>'DT'</B> para la presentación de los datos</li>"))
                ),
                box(title = "Aviso", status = "warning", solidHeader = TRUE, align = "left", width = "adjust",
                    helpText( HTML( "Para una mejor experiencia recomendamos maximizar la ventana y ajustar el zoom pulsando <B>CTRL
                    y +</B> o con <B>CTRL y -</B> .<br><br>
                    Eurostat dispone de una copiosa cantidad de datos, incluso dentro de cada conjunto de datos. <br>
      Es por ello que la carga de estos últimos puede conllevar un moderado tiempo de espera en la primera ejecución de la aplicación. <br>
      Si los datos no se muestran, le rogamos que espere unos segundos. <br><br> Gracias por su comprensión."))
                )
              )
      ),

      # Pestaña de Visión general-----------------------------------------------------------------------------------------------
      tabItem(tabName = "Vision_general",
              fluidRow(
```

```
            box(title = "Descripción", status = "success", solidHeader = TRUE, width = "adjust",  align = "center", collapsible = TRUE,
                helpText(HTML("Esta pestaña muestra los datos más relevantes de los estudios realizados. <br>
              Algunas de las indicaciones son el conjunto de datos del que proviene, su valor más reciente, sus unidades y
              su fecha de actualización"))
            ),
            box(title = "Visión general de la Unión Europea", status = "primary", solidHeader = TRUE, width = "adjust",
                tabsetPanel(id = "vision_panel",
                            tabPanel("PIB y otros indicadores nacionales", dataTableOutput("vision_tabla_PIB")),
                            tabPanel("Consumidores", dataTableOutput("vision_tabla_consumidores")),
                            tabPanel("Empleo y Vacantes", dataTableOutput("vision_tabla_empleo")),
                            tabPanel("Densidad de población en España", dataTableOutput("vision_tabla_densidad")),
                            tabPanel("Objetivos de Desarrollo Sostenible", dataTableOutput("vision_tabla_ODS"))
                )
            )
          )
  ),

  # Pestaña de PIB~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  tabItem(tabName = "PIB",
          fluidRow(
            box(title= "Descripción", status = "success", solidHeader = TRUE, width = "adjust", align = "center", collapsible = TRUE,
                helpText(HTML("Observar la evolución de la Unión Europea y similares mediante indicadores como el PIB <br>
              Los datos son trimestrales <br>
              Las unidades son miles de millones de euros"))
            )
          ),
          fluidRow(
            column(width = 4,
                   box(title = "Controles", status = "warning", solidHeader = TRUE, width = NULL,
                       selectInput(
                         "PIB_variable", label = h4("Variable de estudio"),
                         choices = list("PIB", "Gasto en consumo final",
                                        "Formación bruta de capital fijo" = "Formacion bruta de capital fijo", "Exportaciones",
                                        "Importaciones"),
                         selected = "PIB"),
                       checkboxGroupInput("PIB_geo", label = h4("Área geográfica de estudio"),
                                          choices = list("Unión Europea", "Zona del euro", "España", "Francia", "Alemania", "Reino Unido"),
                                          selected = "Unión Europea")
                   ),
                   box(width = NULL, background = "purple",
                       uiOutput("PIB_fechas")
                   ),
                   infoBoxOutput("PIB_fecha_rango", width = NULL),
                   box(title = "Explicación", status = "info", solidHeader = TRUE, width = NULL, htmlOutput("PIB_explicacion"))
            ),
            column(width = 8,
                   box(
                     title = "Gráfico", status = "primary", solidHeader = TRUE, width = NULL,
                     plotlyOutput("PIB_plot", height = '600px')
                   )
            )
          )
  ),

  # Pestaña de Consumidores~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  tabItem(tabName = "Consumidores",
          fluidRow(
            box(title= "Descripción", status = "success", solidHeader = TRUE, width = "adjust", align = "center", collapsible = TRUE,
                helpText(HTML( "Observar la evolución de los consumidores de la Unión Europea mediante indicadores <br>
                Los datos son mensuales <br>
                Las unidades son puntos de equilibrio. Sirven para comparar la evolución."))
            )
          ),
          fluidRow(
            column(width = 4,
                   box(title = "Controles", status = "warning", solidHeader = TRUE, width = NULL,
                       checkboxGroupInput("consumer_indicador", label = h4("Indicador de estudio"),
                                          choices = list("Indicador de la confianza de los consumidores",
                                                         "Situación financiera para los próximos 12 meses",
                                                         "Situación económica general para los próximos 12 meses",
                                                         "Expectativas de desempleo para los próximos 12 meses",
                                                         "Ahorros para los próximos 12 meses",
                                                         "Tendencias de precios en los últimos 12 meses",
                                                         "Tendencias de precios para los próximos 12 meses"),
                                          selected = "Indicador de la confianza de los consumidores")
                   ),
                   box(width = NULL, background = "purple",
                       uiOutput("consumer_fechas")
                   ),
                   infoBoxOutput("consumer_fecha_rango", width = NULL)
            ),
            column(width = 8,
                   box(
                     title = "Gráfico", status = "primary", solidHeader = TRUE, width = NULL,
                     plotlyOutput("consumer_plot", height = '600px')
                   )
            )
```

```
        )
    ),

    # Pestaña de Empleo~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    tabItem(tabName = "Empleo",
            fluidRow(
              box(title= "Descripción", status = "success", solidHeader = TRUE, width = "adjust", align = "center", collapsible = TRUE,
                  helpText( HTML( "Observar la evolución del desempleo y vacantes de empleo en la Unión Europea <br>
                  Los datos son mensuales para el desempleo y trimestrales para vacantes <br>
                  Las unidades son porcentajes" ) )
              )
            ),
            fluidRow(
              column(width = 4,
                     box(title = "Desempleo - Selección del estudio", status = "warning", solidHeader = TRUE, width = NULL,
                         checkboxGroupInput("unemployment_estudio", label = "",
                                            choices = list("General", "Por género", "Por edad"),
                                            selected = "General")
                     ),
                     box(width = NULL, background = "purple",
                         uiOutput("empleo_fechas")
                     ),
                     infoBoxOutput("unemployment_fecha_rango", width = NULL),
                     infoBoxOutput("vacancy_fecha_rango", width = NULL)
              ),
              column(width = 8,
                     box(
                       title = "Ratio de desempleo", status = "primary", solidHeader = TRUE, width = NULL,
                       plotlyOutput("unemployment_plot", height = '325px')
                     )
              )
            ),
            fluidRow(
              column(width = 4,
                     box(title = "Vacantes - Selección del sector", status = "warning", solidHeader = TRUE, width = NULL,
                         checkboxGroupInput("vacancy_sector", label = "",
                                            choices = list("Economía empresarial", "Industria, construcción y servicios",
                                                           "Fabricación", "Ventas, transporte, alojamiento y servicio de comida",
                                                           "Información y comunicación", "Financiero y seguros", "Inmobiliario",
                                                           "Científico y técnico"),
                                            selected = "Economía empresarial")
                     ),
                     box(title = "Explicación", status = "info", solidHeader = TRUE, width = NULL,
                         helpText(HTML("El <B>ratio de desempleo</B> se calcula como el número de personas en situación de desempleo
                         dividido por el total de la fuerza de trabajo (empleados y desempleados)<br>
                  La <B>tasa de vacantes</B> se calcula como el número de puestos vacantes dividido por el total de puestos
                  (ocupados y vacantes)"))
                     )
              ),
              column(width = 8,
                     box(
                       title = "Tasa de vacantes", status = "primary", solidHeader = TRUE, width = NULL,
                       plotlyOutput("vacancy_plot", height = '325px')
                     )

              )
            )
    ),

    # Pestaña de Densidad de población~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    tabItem(tabName = "Densidad_población",
            fluidRow(
              box(title= "Descripción", status = "success", solidHeader = TRUE, width = "adjust", align = "center", collapsible = TRUE,
                  helpText(HTML( "Observar la densidad de población según el nivel de precisión geográfica <br>
                  Las unidades son habitantes por kilómetro cuadrado <br>"))
              )
            ),
            fluidRow(
              column(width = 4,
                     box(title = "Selección del nivel NUTS", status = "warning", solidHeader = TRUE, width = NULL,
                         selectInput(
                           "density_level", label = "",
                           choices = list("NUTS 1",  "NUTS 2", "NUTS 2", "NUTS 3"),
                           selected = "NUTS 1")
                     ),
                     box(width = NULL, background = "purple",
                         uiOutput("density_fecha")
                     ),
                     infoBoxOutput("density_fecha_rango", width = NULL),
                     box(title = "Explicación de los niveles geográficos", status = "info", solidHeader = TRUE, width = NULL,
                         helpText(HTML("La <B>Nomenclatura de las Unidades Territoriales Estadísticas</B> (NUTS en francés)
                         define demarcaciones territoriales utilizadas por la Unión Europea con fines estadísticos.
                          Veamos el caso de España:<br><br>
                         <B>NUTS 1</B> - Grupos de Comunidades Autónomas (p.ej. 'Noroeste', 'Sur') <br><br>
                          <B>NUTS 2</B> - Comunidades y Ciudades Autónomas (p.ej. 'Galicia', 'Andalucía') <br><br>
                          <B>NUTS 3</B> - Provincias, Consejos insulares y Cabildos (p.ej. 'A Coruña', 'Almería')"))
                     )
```

```
          ),
          column(width = 8,
                 box(title = "Densidad de población", status = "primary", solidHeader = TRUE, width = "adjust",
                     dataTableOutput("density_tabla")
                 )
          )
        )
      ),

      # Pestaña de ODS~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      tabItem(tabName = "ODS",
              fluidRow(
                box(title= "Descripción", status = "success", solidHeader = TRUE, width = "adjust", align = "center", collapsible = TRUE,
                    helpText( HTML( "Observar el cumplimiento de dos Objetivos de Desarrollo sostenible por parte de la UE y distintos países:
                    El <em>Objetivo 5 - Igualdad de género</em> y el <em>Objetivo 13 - Acción por el clima</em> <br>
                    Los datos son anuales <br>
                    Las unidades son porcentaje de la población total y un índice base 100 (en el año 1990)" ) )
                )
              ),
              fluidRow(
                column(width = 4,
                       box(title = "Selección del área geográfica de estudio", status = "warning", solidHeader = TRUE, width = NULL,
                           checkboxGroupInput("ODS_geo", label = "",
                                              choices = list("Unión Europea", "España", "Francia", "Alemania", "Reino Unido"),
                                              selected = "Unión Europea")
                       ),
                       box(width = NULL, background = "purple",
                           uiOutput("ODS_fechas")
                       ),
                       infoBoxOutput("gender_fecha_rango", width = NULL)
                ),
                column(width = 8,
                       box(
                         title = "Brecha de género en la contratación", status = "primary", solidHeader = TRUE, width = NULL,
                         plotlyOutput("gender_plot", height = '325px')
                       )
                )
              ),
              fluidRow(
                column(width = 4,
                       infoBoxOutput("gas_fecha_rango", width = NULL),
                       box(title = "Explicación de los ODS", status = "info", solidHeader = TRUE, width = NULL,
                           helpText(HTML("<B>Brecha de género en la contratación</B>: El indicador mide la diferencia entre los
                           ratios de contratación de hombres y mujeres de edad entre 20 y 64 años.
                           El ratio de contratación se calcula dividiendo el número de personas que trabajan y se encuentran en
                           el rango de edad indicado entre la población total del mismo rango de edad. <br><br>
                                         <B>Emisiones de gases de efecto invernadero</B>: El indicador mide las emisiones nacionales
                           totales, utilizando el potencial de calentamiento global de cada gas para su integración en el indicador"))
                       )
                ),
                column(width = 8,
                       box(
                         title = "Emisiones de gases de efecto invernadero", status = "primary", solidHeader = TRUE, width = NULL,
                         plotlyOutput("gas_plot", height = '325px')
                       )
                )
              )
      )
    )
  )
)

# Define server logic required-------------------------------------------------------------------------------------------------
server <- function(input, output) {

  # Código para la pestaña de PIB~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  PIB_data <- get_eurostat("namq_10_gdp", type = "label") %>%
    filter(s_adj == "Seasonally and calendar adjusted data") %>%
    filter( time >= "2015-01-01" & unit == "Chain linked volumes (2015), million euro"
            | time < "2010-01-01" & unit == "Chain linked volumes (2005), million euro"
            | time >= "2010-01-01" & time < "2015-01-01" & unit == "Chain linked volumes (2010), million euro") %>%
    select(-s_adj, -unit) %>%
    filter(na_item %in% c("Value added, gross", "Final consumption expenditure", "Gross fixed capital formation",
                          "Exports of goods and services", "Imports of goods and services")) %>%
    filter( time >= "2020-02-01" & geo == "European Union - 27 countries (from 2020)" # Salida de Reino Unido
            | time < "2020-02-01" & time >= "2005-01-01" & geo == "European Union - 28 countries (2013-2020)"
            | time < "2005-01-01" & geo == "European Union - 15 countries (1995-2004)"
            | geo %in% c("Spain", "France", "Germany (until 1990 former territory of the FRG)", "United Kingdom",
                         "Euro area (EA11-1999, EA12-2001, EA13-2007, EA15-2008, EA16-2009, EA17-2011, EA18-2014, EA19-2015)") ) %>%
    rename(fecha = time, valores = values, variable = na_item)

  PIB_data$valores <- PIB_data$valores / 1000
  PIB_data$variable <- as.character(PIB_data$variable)
  PIB_data$geo <- as.character(PIB_data$geo)
  PIB_data$variable[ grepl("Value added, gross", PIB_data$variable) ] <- "PIB"
  PIB_data$variable[ grepl("Final consumption expenditure", PIB_data$variable) ] <- "Gasto en consumo final"
  PIB_data$variable[ grepl("Gross fixed capital formation", PIB_data$variable) ] <- "Formacion bruta de capital fijo" #No puede llevar tilde
```

```r
PIB_data$variable[ grepl("Exports of goods and services", PIB_data$variable) ] <- "Exportaciones"
PIB_data$variable[ grepl("Imports of goods and services", PIB_data$variable) ] <- "Importaciones"

PIB_data$geo[ grepl("European Union", PIB_data$geo) ] <- "Unión Europea"
PIB_data$geo[ grepl("Euro area", PIB_data$geo) ] <- "Zona del euro"
PIB_data$geo[ grepl("Spain", PIB_data$geo) ] <- "España"
PIB_data$geo[ grepl("France", PIB_data$geo) ] <- "Francia"
PIB_data$geo[ grepl("United Kingdom", PIB_data$geo) ] <- "Reino Unido"
PIB_data$geo[ grepl("Germany", PIB_data$geo) ] <- "Alemania"

output$PIB_fechas <-renderUI(
  sliderInput("PIB_slider", label = h4("Rango de fechas"), timeFormat = '%Y-%m',
              min = min(PIB_data$fecha), max = max(PIB_data$fecha),
              value = c(min(PIB_data$fecha), max(PIB_data$fecha)))
)

output$PIB_plot <- renderPlotly({

  PIB_fecha_min <- input$PIB_slider[1]
  PIB_fecha_max <- input$PIB_slider[2]

  check_EU <- "Unión Europea" %in% input$PIB_geo
  check_eurozone <- "Zona del euro" %in% input$PIB_geo
  check_spain <- "España" %in% input$PIB_geo
  check_france <- "Francia" %in% input$PIB_geo
  check_germany <- "Alemania" %in% input$PIB_geo
  check_UK <- "Reino Unido" %in% input$PIB_geo

  ggplotly(
    ggplot(PIB_data)  +
      {if(check_EU) geom_line(PIB_data %>%
          filter(geo == "Unión Europea" & variable == input$PIB_variable), mapping = aes(x = fecha, y = valores, color = geo))} +
      {if(check_eurozone) geom_line(PIB_data %>%
          filter(geo == "Zona del euro" & variable == input$PIB_variable), mapping = aes(x = fecha, y = valores, color = geo))} +
      {if(check_spain) geom_line(PIB_data %>%
          filter(geo == "España" & variable == input$PIB_variable), mapping = aes(x = fecha, y = valores, color = geo))} +
      {if(check_france) geom_line(PIB_data %>%
          filter(geo == "Francia" & variable == input$PIB_variable), mapping = aes(x = fecha, y = valores, color = geo))} +
      {if(check_germany) geom_line(PIB_data %>%
          filter(geo == "Alemania" & variable == input$PIB_variable), mapping = aes(x = fecha, y = valores, color = geo))} +
      {if(check_UK) geom_line(PIB_data %>%
          filter(geo == "Reino Unido" & variable == input$PIB_variable), mapping = aes(x = fecha, y = valores, color = geo))} +
      scale_x_date(limits = c(PIB_fecha_min, PIB_fecha_max), date_labels = "%Y", date_breaks = "5 years") +
      theme(legend.title = element_blank())
  ) %>%
    add_annotations( text = "Geo", xref="paper", yref="paper", x=1.04, xanchor="left", y=0.8, yanchor="bottom", legendtitle = TRUE,
                     showarrow=FALSE) %>%
    layout(legend = list(y = 0.8, yanchor="top"))
})

output$PIB_fecha_rango <- renderInfoBox(
  infoBox(HTML("Rango de fechas de los datos"), paste(format(min(PIB_data$fecha), '%Y-%m'), "hasta",
                                                    format(max(PIB_data$fecha), '%Y-%m')), icon = icon("calendar"), color = "purple")
)

output$PIB_explicacion <- renderText(
  switch (input$PIB_variable,
          "PIB" = paste( "El <B>Producto Interior Bruto</B> es un indicador económico que refleja el valor monetario de todos los
          bienes y servicios finales producidos por un país o región en un determinado periodo de tiempo, normalmente un año.
          Se utiliza para medir la riqueza de un país."),
          "Gasto en consumo final" = paste("El <B>gasto en consumo final</B> consiste en el gasto realizado por las unidades
          institucionales residentes en bienes y servicios que se utilizan para satisfacer directamente las necesidades o carencias
          individuales o las necesidades colectivas de los miembros de la comunidad."),
          "Formacion bruta de capital fijo" = paste("Se denomina <B>formación bruta de capital fijo</B> a la medición de los aumentos
          o disminuciones de la cantidad total de bienes duraderos en cierto territorio durante un tiempo en cuestión."),
          "Exportaciones" = paste("Las <B>exportaciones de bienes y servicios</B> consisten en operaciones (ventas, regalos o
          donaciones) mediante las cuales los residentes suministran bienes y servicios a los no residentes."),
          "Importaciones" = paste("Las <B>importaciones de bienes y servicios</B> consisten en operaciones (adquisiciones, trueques,
          regalos o donaciones) mediante las cuales los no residentes suministran bienes y servicios a los residentes.")
  )
)


# Código para la pestaña de Consumidores ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
consumer_data <- get_eurostat("ei_bsco_m", type = "label") %>%
  filter(grepl("Seasonally", s_adj)) %>%
  filter(grepl("European Union", geo)) %>%
  select(indic, time, values) %>%
  filter(indic %in% c("Consumer confidence indicator", "Financial situation over the next 12 months", "General economic situation over
                    the next 12 months",
                    "Unemployment expectations over the next 12 months", "Savings over the next 12 months",
                    "Price trends over the last 12 months", "Price trends over the next 12 months")) %>%
  rename(indicador = indic, fecha = time, valores = values)

consumer_data$indicador <- as.character(consumer_data$indicador)
consumer_data$indicador[consumer_data$indicador == "Consumer confidence indicator"] <- "Indicador de la confianza de los consumidores"
consumer_data$indicador[consumer_data$indicador == "Financial situation over the next 12 months"] <- "Situación financiera para los
```

```r
                                                    próximos 12 meses"
consumer_data$indicador[consumer_data$indicador == "General economic situation over the next 12 months"] <- "Situación económica
                                                    general para los próximos 12 meses"
consumer_data$indicador[consumer_data$indicador == "Unemployment expectations over the next 12 months"] <- "Expectativas de desempleo
                                                    para los próximos 12 meses"
consumer_data$indicador[consumer_data$indicador == "Savings over the next 12 months"] <- "Ahorros para los próximos 12 meses"
consumer_data$indicador[consumer_data$indicador == "Price trends over the last 12 months"] <- "Tendencias de precios en los últimos 12 meses"
consumer_data$indicador[consumer_data$indicador == "Price trends over the next 12 months"] <- "Tendencias de precios para los próximos 12 meses"

output$consumer_fechas <-renderUI(
  sliderInput("consumer_slider", label = h4("Rango de fechas"), timeFormat = '%Y-%m',
              min = min(consumer_data$fecha), max = max(consumer_data$fecha),
              value = c(min(consumer_data$fecha), max(consumer_data$fecha)))
)

output$consumer_plot <- renderPlotly({

  consumer_fecha_min <- input$consumer_slider[1]
  consumer_fecha_max <- input$consumer_slider[2]

  check_confidence <- "Indicador de la confianza de los consumidores" %in% input$consumer_indicador
  check_financial <- "Situación financiera para los próximos 12 meses" %in% input$consumer_indicador
  check_general <- "Situación económica general para los próximos 12 meses" %in% input$consumer_indicador
  check_unemployment <- "Expectativas de desempleo para los próximos 12 meses" %in% input$consumer_indicador
  check_savings <- "Ahorros para los próximos 12 meses" %in% input$consumer_indicador
  check_price_last <- "Tendencias de precios en los últimos 12 meses" %in% input$consumer_indicador
  check_price_next <- "Tendencias de precios para los próximos 12 meses" %in% input$consumer_indicador

  ggplotly(
    ggplot(consumer_data) +
      {if(check_confidence) geom_line(consumer_data %>%
        filter(indicador == "Indicador de la confianza de los consumidores"), mapping = aes(x = fecha, y = valores, color = indicador))} +
      {if(check_financial) geom_line(consumer_data %>%
        filter(indicador == "Situación financiera para los próximos 12 meses"), mapping = aes(x = fecha, y = valores, color = indicador))} +
      {if(check_general) geom_line(consumer_data %>%
        filter(indicador == "Situación económica general para los próximos 12 meses"), mapping = aes(x = fecha, y = valores, color = indicador))} +
      {if(check_unemployment) geom_line(consumer_data %>%
        filter(indicador == "Expectativas de desempleo para los próximos 12 meses"), mapping = aes(x = fecha, y = valores, color = indicador))} +
      {if(check_savings) geom_line(consumer_data %>%
        filter(indicador == "Ahorros para los próximos 12 meses"), mapping = aes(x = fecha, y = valores, color = indicador))} +
      {if(check_price_last) geom_line(consumer_data %>%
        filter(indicador == "Tendencias de precios en los últimos 12 meses"), mapping = aes(x = fecha, y = valores, color = indicador))} +
      {if(check_price_next) geom_line(consumer_data %>%
        filter(indicador == "Tendencias de precios para los próximos 12 meses"), mapping = aes(x = fecha, y = valores, color = indicador))} +
      scale_x_date(limits = c(consumer_fecha_min, consumer_fecha_max), date_labels = "%Y") +
      theme(legend.title = element_blank())
  ) %>%
    add_annotations( text = "Indicadores", xref="paper", yref="paper", x=0.2, y=-0.4, xanchor="center", yanchor="top", legendtitle=TRUE,
                     showarrow=FALSE) %>%
    layout(legend = list(x = 0.5, y = -0.4, xanchor="center", yanchor = "top"))
})

output$consumer_fecha_rango <- renderInfoBox(
  infoBox(HTML("Rango de fechas de los datos"), paste(format(min(consumer_data$fecha), '%Y-%m'), "hasta",
                                                      format(max(consumer_data$fecha), '%Y-%m')), icon = icon("calendar"), color = "purple")
)


# Código para la pestaña de Empleo ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
unemployment_data <- get_eurostat("ei_lmhr_m", type = "label") %>%
  filter(grepl("Seasonally adjusted data", s_adj)) %>%
  select(-unit, -s_adj) %>%
  filter( grepl("European Union", geo)
          & time >= "2020-02-01" & geo == "European Union - 27 countries (from 2020)" # Salida de Reino Unido
          | time < "2020-02-01" & time >= "2013-07-01" & geo == "European Union - 28 countries (2013-2020)" # Entrada de Croacia
          | time < "2013-07-01" & geo == "European Union - 27 countries (2007-2013)") %>%
  rename(indicador = indic, fecha = time, valores = values) %>%
  select(-geo)

unemployment_data$indicador <- as.character(unemployment_data$indicador)
unemployment_data$indicador[ grepl("Over 25 years - Females", unemployment_data$indicador) ] <- "Desempleo entre mujeres de más de 25 años"
unemployment_data$indicador[ grepl("Under 25 years - Females", unemployment_data$indicador) ] <- "Desempleo entre mujeres de menos de 25 años"
unemployment_data$indicador[ grepl("definition - Females", unemployment_data$indicador) ] <- "Desempleo entre mujeres"
unemployment_data$indicador[ grepl("Over 25 years - Males", unemployment_data$indicador) ] <- "Desempleo entre hombres de más de 25 años"
unemployment_data$indicador[ grepl("Under 25 years - Males", unemployment_data$indicador) ] <- "Desempleo entre hombres de menos de 25 años"
unemployment_data$indicador[ grepl("definition - Males", unemployment_data$indicador) ] <- "Desempleo entre hombres"
unemployment_data$indicador[ grepl("Over 25 years - Total", unemployment_data$indicador) ] <- "Desempleo general entre personas de más de 25 años"
unemployment_data$indicador[ grepl("Under 25 years - Total", unemployment_data$indicador) ] <- "Desempleo general entre personas de menos de 25 años"
unemployment_data$indicador[ grepl("definition - Total", unemployment_data$indicador) ] <- "Desempleo general"

output$empleo_fechas <-renderUI(
  sliderInput("empleo_slider", label = h4("Rango de fechas para Desempleo y Vacantes"), timeFormat = '%Y-%m',
              min = min(unemployment_data$fecha), max = max(unemployment_data$fecha),
              value = c(min(unemployment_data$fecha), max(unemployment_data$fecha)))
)

output$unemployment_plot <- renderPlotly({
```

```
  unemployment_fecha_min <- input$empleo_slider[1]
  unemployment_fecha_max <- input$empleo_slider[2]

  check_general <- "General" %in% input$unemployment_estudio
  check_gender <- "Por género" %in% input$unemployment_estudio
  check_age <- "Por edad" %in% input$unemployment_estudio

  ggplotly(
    ggplot(unemployment_data)  +
      {if(check_general) geom_line(unemployment_data %>%
        filter(indicador == "Desempleo general"), mapping = aes(x = fecha, y = valores, color = indicador))} +
      {if(check_gender & !check_age) geom_line(unemployment_data %>%
        filter(indicador %in% c("Desempleo entre mujeres", "Desempleo entre hombres")), mapping = aes(x = fecha, y = valores,
                                                                                                      color = indicador))} +
      {if(check_age & check_gender) geom_line(unemployment_data %>%
        filter(indicador %in% c("Desempleo entre mujeres de más de 25 años", "Desempleo entre mujeres de menos de 25 años",
                                "Desempleo entre hombres de más de 25 años", "Desempleo entre hombres de menos de 25 años")),
          mapping = aes(x = fecha, y = valores, color = indicador))} +
      {if(check_age & !check_gender) geom_line(unemployment_data %>%
        filter(grepl("personas", indicador)), mapping = aes(x = fecha, y = valores, color = indicador))} +
      scale_x_date(limits = c(unemployment_fecha_min, unemployment_fecha_max),date_labels = "%Y") +
      theme(legend.title = element_blank())
  ) %>%
    add_annotations(text = "Estudio", xref="paper", yref="paper", x=1.04, xanchor="left", y=0.8, yanchor="bottom", legendtitle = TRUE,
                    showarrow=FALSE) %>%
    layout(legend = list(y = 0.8, yanchor="top"))
})

output$unemployment_fecha_rango <- renderInfoBox(
  infoBox(HTML("Rango de fechas para <B>DESEMPLEO</B>"), paste(format(min(unemployment_data$fecha), '%Y-%m'), "hasta",
                                                               format(max(unemployment_data$fecha), '%Y-%m')), icon = icon("calendar"),
                                                                color = "purple")
)

vacancy_data <- get_eurostat("ei_lmjv_q_r2", type = "label") %>%
  filter(grepl("Unadjusted", s_adj)) %>% #Sin ajuste porque para la UE con ajuste solo hay 2 sectores
  filter(sizeclas == "Total") %>%
  filter( nace_r2 != "Industry (except construction)" & nace_r2 != "Construction") %>%
  select(-s_adj, -sizeclas, -indic) %>%
  filter( time >= "2020-02-01" & geo == "European Union - 27 countries (from 2020)"
          | time < "2020-02-01" & geo == "European Union - 28 countries (2013-2020)") %>%
  select(-geo) %>%
  rename(sector = nace_r2, fecha = time, valores = values)

vacancy_data$sector <- as.character(vacancy_data$sector)
vacancy_data$sector[ grepl("Industry,", vacancy_data$sector) ] <- "Industria, construcción y servicios"
vacancy_data$sector[ grepl("Business", vacancy_data$sector) ] <- "Economía empresarial"
vacancy_data$sector[ grepl("Manufacturing", vacancy_data$sector) ] <- "Fabricación"
vacancy_data$sector[ grepl("Wholesale", vacancy_data$sector) ] <- "Ventas, transporte, alojamiento y servicio de comida"
vacancy_data$sector[ grepl("Information", vacancy_data$sector) ] <- "Información y comunicación"
vacancy_data$sector[ grepl("Financial", vacancy_data$sector) ] <- "Financiero y seguros"
vacancy_data$sector[ grepl("Real estate", vacancy_data$sector) ] <- "Inmobiliario"
vacancy_data$sector[ grepl("Professional", vacancy_data$sector) ] <- "Científico y técnico"

output$vacancy_plot <- renderPlotly({

  vacancy_fecha_min <- input$empleo_slider[1]
  vacancy_fecha_max <- input$empleo_slider[2]

  check_industry <- "Industria, construcción y servicios" %in% input$vacancy_sector
  check_business <- "Economía empresarial" %in% input$vacancy_sector
  check_manufacturing <- "Fabricación" %in% input$vacancy_sector
  check_wholesale <- "Ventas, transporte, alojamiento y servicio de comida" %in% input$vacancy_sector
  check_information <- "Información y comunicación" %in% input$vacancy_sector
  check_financial <- "Financiero y seguros" %in% input$vacancy_sector
  check_realestate <- "Inmobiliario" %in% input$vacancy_sector
  check_professional <- "Científico y técnico" %in% input$vacancy_sector

  ggplotly(
    ggplot(vacancy_data)  +
      {if(check_business) geom_line(vacancy_data %>%
        filter(sector == "Economía empresarial"), mapping = aes(x = fecha, y = valores, color = sector))} +
      {if(check_industry) geom_line(vacancy_data %>%
        filter(sector == "Industria, construcción y servicios"), mapping = aes(x = fecha, y = valores, color = sector))} +
      {if(check_manufacturing) geom_line(vacancy_data %>%
        filter(sector == "Fabricación"), mapping = aes(x = fecha, y = valores, color = sector))} +
      {if(check_wholesale) geom_line(vacancy_data %>%
        filter(sector == "Ventas, transporte, alojamiento y servicio de comida"), mapping = aes(x = fecha, y = valores, color = sector))} +
      {if(check_information) geom_line(vacancy_data %>%
        filter(sector == "Información y comunicación"), mapping = aes(x = fecha, y = valores, color = sector))} +
      {if(check_financial) geom_line(vacancy_data %>%
        filter(sector == "Financiero y seguros"), mapping = aes(x = fecha, y = valores, color = sector))} +
      {if(check_realestate) geom_line(vacancy_data %>%
        filter(sector == "Inmobiliario"), mapping = aes(x = fecha, y = valores, color = sector))} +
      {if(check_professional) geom_line(vacancy_data %>%
        filter(sector == "Científico y técnico"), mapping = aes(x = fecha, y = valores, color = sector))} +
      scale_x_date(limits = c(vacancy_fecha_min, vacancy_fecha_max), date_labels = "%Y") +
```

```
      theme(legend.title = element_blank())
  ) %>%
    add_annotations(text = "Sector", xref="paper", yref="paper", x=1.04, xanchor="left", y=0.8, yanchor="bottom", legendtitle = TRUE,
                    showarrow=FALSE) %>%
    layout(legend = list(y = 0.8, yanchor="top"))
})

output$vacancy_fecha_rango <- renderInfoBox(
  infoBox(HTML("Rango de fechas para <B>VACANTES</B>"), paste(format(min(vacancy_data$fecha), '%Y-%m'), "hasta",
                                                              format(max(vacancy_data$fecha), '%Y-%m')), icon = icon("calendar"),
                                                      color = "purple")
)


# Código para la pestaña de Densidad población ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
density_data <- get_eurostat("demo_r_d3dens") %>%
  filter(grepl("ES", geo))

geo_label <- density_data %>%
  label_eurostat(fix_duplicated = TRUE) %>%
  select(geo) %>%
  rename(area = geo)

geo_label$area <- as.character(geo_label$area)
geo_label$area[ grepl("Spain", geo_label$area) ] <- "España"
geo_label$area[ grepl("Cantabria", geo_label$area) ] <- "Cantabria"
geo_label$area[ grepl("La Rioja", geo_label$area) ] <- "La Rioja"
geo_label$area[ grepl("Comunidad de Madrid", geo_label$area) ] <- "Comunidad de Madrid"
geo_label$area[ grepl("Canarias", geo_label$area) ] <- "Canarias"

density_data <- density_data %>%
  select(-unit) %>%
  rename(fecha = time, densidad = values)

density_data["area"] <- geo_label$area
density_data <- density_data[c("geo", "area", "fecha", "densidad")]

density_nuts1 <- density_data %>%
  filter( geo == "ES" | str_length(geo) == 3 )

density_nuts2 <- density_data %>%
  filter(  geo == "ES" | str_length(geo) == 4 )

density_nuts3 <- density_data %>%
  filter(  geo == "ES" | str_length(geo) == 5 )

output$density_fecha <-renderUI(
  sliderInput("density_slider", label = h4("Año de estudio"), timeFormat = '%Y',
              min = min(density_data$fecha), max = max(density_data$fecha),
              value = max(density_data$fecha))
)

output$density_fecha_rango <- renderInfoBox(
  infoBox(HTML("Rango de fechas"), paste(format(min(density_data$fecha), '%Y'), "hasta",
                                         format(max(density_data$fecha), '%Y')), icon = icon("calendar"), color = "purple")
)



output$density_tabla <- renderDataTable({

  check_1 <- grepl("NUTS 1", input$density_level)
  check_2 <- grepl("NUTS 2", input$density_level)
  check_3 <- grepl("NUTS 3", input$density_level)

  año_estudio <- floor_date(input$density_slider, "year")

  if(check_1) density_t <- data.frame(
    Código = pull(density_nuts1 %>% filter(fecha == año_estudio) %>% select(geo)),
    Área = pull(density_nuts1 %>% filter(fecha == año_estudio) %>% select(area)),
    Densidad = pull(density_nuts1 %>% filter(fecha == año_estudio) %>% select(densidad))
  )

  if(check_2) density_t <- data.frame(
    Código = pull(density_nuts2 %>% filter(fecha == año_estudio) %>% select(geo)),
    Área = pull(density_nuts2 %>% filter(fecha == año_estudio) %>% select(area)),
    Densidad = pull(density_nuts2 %>% filter(fecha == año_estudio) %>% select(densidad))
  )

  if(check_3) density_t <- data.frame(
    Código = pull(density_nuts3 %>% filter(fecha == año_estudio) %>% select(geo)),
    Área = pull(density_nuts3 %>% filter(fecha == año_estudio) %>% select(area)),
    Densidad = pull(density_nuts3 %>% filter(fecha == año_estudio) %>% select(densidad))
  )

  datatable(density_t, rownames = FALSE, options = list(dom = 'ftp', orderClasses = TRUE,
                                        language = list(paginate = list('next' = "Siguiente", previous = "Anterior"),
```

```
                                        search = "Buscar:", zeroRecords = "Sin resultados de búsqueda"),
                                        initComplete = JS(
                                          "function(settings, json) {",
                                          "$(this.api().table().header()).css({'background-color': '#6495ED', 'color': '#F8F8FF'});",
                                          "}")))
}
  )

  # Código para la pestaña de ODS ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  gender_data <- get_eurostat("sdg_05_30", type = "label") %>%
    select(geo, time, values) %>%
    filter(time >= "2020-02-01" & geo == "European Union - 27 countries (from 2020)" # Salida de Reino Unido
           | time < "2020-02-01" & geo == "European Union - 28 countries (2013-2020)"
           | geo %in% c("Spain", "France", "Germany (until 1990 former territory of the FRG)", "United Kingdom")) %>%
    rename(fecha = time, brecha = values) %>%
    arrange(desc(fecha))

  gender_data$geo <- as.character(gender_data$geo)
  gender_data$geo[ grepl("European Union", gender_data$geo) ] <- "Unión Europea"
  gender_data$geo[ grepl("Spain", gender_data$geo) ] <- "España"
  gender_data$geo[ grepl("France", gender_data$geo) ] <- "Francia"
  gender_data$geo[ grepl("Germany", gender_data$geo) ] <- "Alemania"
  gender_data$geo[ grepl("United Kingdom", gender_data$geo) ] <- "Reino Unido"


  gas_data <- get_eurostat("sdg_13_10", type = "label") %>%
    filter(time >= "2020-02-01" & geo == "European Union - 27 countries (from 2020)" # Salida de Reino Unido
           | time < "2020-02-01" & geo == "European Union - 28 countries (2013-2020)"
           | geo %in% c("Spain", "France", "Germany (until 1990 former territory of the FRG)", "United Kingdom")) %>%
    filter(grepl("CO2", indic_env)) %>%
    select(geo, time, values) %>%
    rename(fecha = time, emisiones = values) %>%
    arrange(desc(fecha))

  gas_data$geo <- as.character(gas_data$geo)
  gas_data$geo[ grepl("European Union", gas_data$geo) ] <- "Unión Europea"
  gas_data$geo[ grepl("Spain", gas_data$geo) ] <- "España"
  gas_data$geo[ grepl("France", gas_data$geo) ] <- "Francia"
  gas_data$geo[ grepl("Germany", gas_data$geo) ] <- "Alemania"
  gas_data$geo[ grepl("United Kingdom", gas_data$geo) ] <- "Reino Unido"


  output$ODS_fechas <-renderUI(
    sliderInput("ODS_slider", label = h4("Rango de fechas"), timeFormat = '%Y',
                min = min(gender_data$fecha, gas_data$fecha), max = max(gender_data$fecha, gas_data$fecha),
                value = c(min(gender_data$fecha, gas_data$fecha), max(gender_data$fecha, gas_data$fecha)))
  )

  output$gender_plot <- renderPlotly({

    ODS_fecha_min <- input$ODS_slider[1]
    ODS_fecha_max <- input$ODS_slider[2]

    check_EU <- "Unión Europea" %in% input$ODS_geo
    check_spain <- "España" %in% input$ODS_geo
    check_france <- "Francia" %in% input$ODS_geo
    check_germany <- "Alemania" %in% input$ODS_geo
    check_UK <- "Reino Unido" %in% input$ODS_geo

    ggplotly(
      ggplot(gender_data)  +
        {if(check_EU) geom_line(gender_data %>%
                                  filter(geo == "Unión Europea"), mapping = aes(x = fecha, y = brecha, color = geo))} +
        {if(check_spain) geom_line(gender_data %>%
                                     filter(geo == "España"), mapping = aes(x = fecha, y = brecha, color = geo))} +
        {if(check_france) geom_line(gender_data %>%
                                      filter(geo == "Francia"), mapping = aes(x = fecha, y = brecha, color = geo))} +
        {if(check_germany) geom_line(gender_data %>%
                                       filter(geo == "Alemania"), mapping = aes(x = fecha, y = brecha, color = geo))} +
        {if(check_UK) geom_line(gender_data %>%
                                  filter(geo == "Reino Unido"), mapping = aes(x = fecha, y = brecha, color = geo))} +
        scale_x_date(limits = c(ODS_fecha_min, ODS_fecha_max), date_labels = "%Y") +
        theme(legend.title = element_blank())
    ) %>%
      add_annotations( text = "Geo", xref="paper", yref="paper", x=1.04, xanchor="left", y=0.8, yanchor="bottom", legendtitle = TRUE,
                       showarrow=FALSE) %>%
      layout(legend = list(y = 0.8, yanchor="top"))
  })

  output$gender_fecha_rango <- renderInfoBox(
    infoBox(HTML("Rango de fechas para <B>BRECHA<B>"), paste(format(min(gender_data$fecha), '%Y'), "hasta",
                                                              format(max(gender_data$fecha), '%Y')), icon = icon("calendar"),
                                                       color = "purple")
  )

  output$gas_plot <- renderPlotly({
```

```r
  ODS_fecha_min <- input$ODS_slider[1]
  ODS_fecha_max <- input$ODS_slider[2]

  check_EU <- "Unión Europea" %in% input$ODS_geo
  check_spain <- "España" %in% input$ODS_geo
  check_france <- "Francia" %in% input$ODS_geo
  check_germany <- "Alemania" %in% input$ODS_geo
  check_UK <- "Reino Unido" %in% input$ODS_geo

  ggplotly(
    ggplot(gas_data)  +
      {if(check_EU) geom_line(gas_data %>%
                              filter(geo == "Unión Europea"), mapping = aes(x = fecha, y = emisiones, color = geo))} +
      {if(check_spain) geom_line(gas_data %>%
                                 filter(geo == "España"), mapping = aes(x = fecha, y = emisiones, color = geo))} +
      {if(check_france) geom_line(gas_data %>%
                                  filter(geo == "Francia"), mapping = aes(x = fecha, y = emisiones, color = geo))} +
      {if(check_germany) geom_line(gas_data %>%
                                   filter(geo == "Alemania"), mapping = aes(x = fecha, y = emisiones, color = geo))} +
      {if(check_UK) geom_line(gas_data %>%
                              filter(geo == "Reino Unido"), mapping = aes(x = fecha, y = emisiones, color = geo))} +
      scale_x_date(limits = c(ODS_fecha_min, ODS_fecha_max), date_labels = "%Y") +
      theme(legend.title = element_blank())
  ) %>%
    add_annotations( text = "Geo", xref="paper", yref="paper", x=1.04, xanchor="left", y=0.8, yanchor="bottom", legendtitle = TRUE,
                     showarrow=FALSE) %>%
    layout(legend = list(y = 0.8, yanchor="top"))
})

output$gas_fecha_rango <- renderInfoBox(
  infoBox(HTML("Rango de fechas para <B>EMISIONES<B>"), paste(format(min(gas_data$fecha), '%Y'), "hasta",
                                                              format(max(gas_data$fecha), '%Y')), icon = icon("calendar"),
                                                        color = "purple")
)



# Código para la pestaña de Visión general~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
v_PIB_PIB <- filter(PIB_data, variable == "PIB" & geo == "Unión Europea")
v_PIB_gcf <- filter(PIB_data, variable == "Gasto en consumo final" & geo == "Unión Europea")
v_PIB_fbcf <- filter(PIB_data, variable == "Formacion bruta de capital fijo" & geo == "Unión Europea")
v_PIB_exp <- filter(PIB_data, variable == "Exportaciones" & geo == "Unión Europea")
v_PIB_imp <- filter(PIB_data, variable == "Importaciones" & geo == "Unión Europea")

v_con_conf <- filter(consumer_data, indicador == "Indicador de la confianza de los consumidores")
v_con_fin <- filter(consumer_data, indicador == "Situación financiera para los próximos 12 meses")
v_con_eco <- filter(consumer_data, indicador == "Situación económica general para los próximos 12 meses")
v_con_desem <- filter(consumer_data, indicador == "Expectativas de desempleo para los próximos 12 meses")
v_con_aho <- filter(consumer_data, indicador == "Ahorros para los próximos 12 meses")
v_con_precio1 <- filter(consumer_data, indicador == "Tendencias de precios en los últimos 12 meses")
v_con_precio2 <- filter(consumer_data, indicador == "Tendencias de precios para los próximos 12 meses")

v_emp_des1 <- filter(unemployment_data, indicador == "Desempleo general")
v_emp_des2 <- filter(unemployment_data, indicador == "Desempleo entre hombres")
v_emp_des3 <- filter(unemployment_data, indicador == "Desempleo entre mujeres")
v_vac_eco <- filter(vacancy_data, sector == "Economía empresarial")
v_vac_cie <- filter(vacancy_data, sector == "Científico y técnico")
v_vac_info <- filter(vacancy_data, sector == "Información y comunicación")

v_dens_esp <- filter(density_data, geo == "ES")
v_dens_noes <- filter(density_data, geo == "ES1")
v_dens_gal <- filter(density_data, geo == "ES11")
v_dens_coru <- filter(density_data, geo == "ES111")
v_dens_este <- filter(density_data, geo == "ES5")
v_dens_comval <- filter(density_data, geo == "ES52")
v_dens_val <- filter(density_data, geo == "ES523")

v_ODS_brechaEU <- filter(gender_data, geo == "Unión Europea")
v_ODS_brechaES <- filter(gender_data, geo == "España")
v_ODS_brechaFR <- filter(gender_data, geo == "Francia")
v_ODS_gasEU <- filter(gas_data, geo == "Unión Europea")
v_ODS_gasES <- filter(gas_data, geo == "España")
v_ODS_gasFR <- filter(gas_data, geo == "Francia")

vision_PIB <- data.frame(
  Variable = c("PIB", "Gasto en consumo final", "Formación bruta de capital fijo", "Exportaciones", "Importaciones"),
  Valor = c(v_PIB_PIB$valores[1], v_PIB_gcf$valores[1], v_PIB_fbcf$valores[1], v_PIB_exp$valores[1], v_PIB_imp$valores[1]),
  Unidades = rep("Miles de millones de euros", 5),
  Fecha = c(v_PIB_PIB$fecha[1], v_PIB_gcf$fecha[1], v_PIB_fbcf$fecha[1], v_PIB_exp$fecha[1], v_PIB_exp$fecha[1])
)
vision_consumidores <- data.frame(
  Variable = c("Indicador de la confianza de los consumidores", "Situación financiera para los próximos 12 meses",
               "Situación económica general para los próximos 12 meses", "Expectativas de desempleo para los próximos 12 meses",
               "Ahorros para los próximos 12 meses", "Tendencias de precios en los últimos 12 meses",
               "Tendencias de precios para los próximos 12 meses"),
  Valor = c(v_con_conf$valores[1], v_con_fin$valores[1], v_con_eco$valores[1],
            v_con_desem$valores[1], v_con_aho$valores[1], v_con_precio1$valores[1], v_con_precio2$valores[1]),
```

```
  Unidades = rep("Unidades de equilibrio", 7),
  Fecha = c(v_con_conf$fecha[1], v_con_fin$fecha[1], v_con_eco$fecha[1],
            v_con_desem$fecha[1], v_con_aho$fecha[1], v_con_precio1$fecha[1], v_con_precio2$fecha[1] )
)
vision_empleo <- data.frame(
  Variable = c("Desempleo general", "Desempleo entre hombres", "Desempleo entre mujeres",
               "Vacantes - Sector de economía empresarial", "Vacantes - Sector científico y técnico",
               "Vacantes - Sector de la información y comunicación"),
  Valor = c(v_emp_des1$valores[1], v_emp_des2$valores[1], v_emp_des3$valores[1],
            v_vac_eco$valores[1], v_vac_cie$valores[1], v_vac_info$valores[1]),
  Unidades = rep("Porcentaje", 6),
  Fecha = c(v_emp_des1$fecha[1], v_emp_des2$fecha[1], v_emp_des3$fecha[1],
            v_vac_eco$fecha[1], v_vac_cie$fecha[1], v_vac_info$fecha[1])
)
vision_densidad <- data.frame(
  'Código NUTS' = c(as.character(v_dens_esp$geo[1]), as.character(v_dens_noes$geo[1]), as.character(v_dens_gal$geo[1]),
                    as.character(v_dens_coru$geo[1]), as.character(v_dens_este$geo[1]), as.character(v_dens_comval$geo[1]),
                    as.character(v_dens_val$geo[1])),
  Área = c(v_dens_esp$area[1], v_dens_noes$area[1], v_dens_gal$area[1], v_dens_coru$area[1],
           v_dens_este$area[1], v_dens_comval$area[1], v_dens_val$area[1]),
  Densidad = c(v_dens_esp$densidad[1], v_dens_noes$densidad[1], v_dens_gal$densidad[1], v_dens_coru$densidad[1],
               v_dens_este$densidad[1], v_dens_comval$densidad[1], v_dens_val$densidad[1]),
  Unidades = rep("Habitantes por kilómetro cuadrado", 7),
  Fecha = c(v_dens_esp$fecha[1], v_dens_noes$fecha[1], v_dens_gal$fecha[1], v_dens_coru$fecha[1],
            v_dens_este$fecha[1], v_dens_comval$fecha[1], v_dens_val$fecha[1])
)
vision_ODS <- data.frame(
  Área = c("Unión Europea", "España", "Francia",
           "Unión Europea", "España", "Francia"),
  Variable = c("Brecha de género en la contratación", "Brecha de género en la contratación", "Brecha de género en la contratación",
               "Emisiones de gas de efecto invernadero", "Emisiones de gas de efecto invernadero",
               "Emisiones de gas de efecto invernadero"),
  Valor = c(v_ODS_brechaEU$brecha[1], v_ODS_brechaES$brecha[1], v_ODS_brechaFR$brecha[1],
            v_ODS_gasEU$emisiones[1], v_ODS_gasES$emisiones[1], v_ODS_gasFR$emisiones[1]),
  Unidades = c("Porcentaje de la población total", "Porcentaje de la población total", "Porcentaje de la población total",
               "Índice base 100 (en el año 1990)", "Índice base 100 (en el año 1990)", "Índice base 100 (en el año 1990)"),
  Fecha = c(v_ODS_brechaEU$fecha[1], v_ODS_brechaES$fecha[1], v_ODS_brechaFR$fecha[1],
            v_ODS_gasEU$fecha[1], v_ODS_gasES$fecha[1], v_ODS_gasFR$fecha[1])
)


output$vision_tabla_PIB <- renderDataTable(
  datatable(vision_PIB, rownames = FALSE, options = list(dom = 'ft', orderClasses = TRUE,
                                      language = list(paginate = list('next' = "Siguiente", previous = "Anterior"),
                                                      search = "Buscar:", zeroRecords = "Sin resultados de búsqueda"),
                                      initComplete = JS(
                                        "function(settings, json) {",
                                        "$(this.api().table().header()).css({'background-color': '#6495ED', 'color': '#F8F8FF'});",
                                        "}")))
)
output$vision_tabla_consumidores <- renderDataTable(
  datatable(vision_consumidores, rownames = FALSE, options = list(dom = 'ft', orderClasses = TRUE,
                                      language = list(paginate = list('next' = "Siguiente", previous = "Anterior"),
                                                      search = "Buscar:", zeroRecords = "Sin resultados de búsqueda"),
                                      initComplete = JS(
                                        "function(settings, json) {",
                                        "$(this.api().table().header()).css({'background-color': '#6495ED', 'color': '#F8F8FF'});",
                                        "}")))
)
output$vision_tabla_empleo <- renderDataTable(
  datatable(vision_empleo, rownames = FALSE, options = list(dom = 'ft', orderClasses = TRUE,
                                      language = list(paginate = list('next' = "Siguiente", previous = "Anterior"),
                                                      search = "Buscar:", zeroRecords = "Sin resultados de búsqueda"),
                                      initComplete = JS(
                                        "function(settings, json) {",
                                        "$(this.api().table().header()).css({'background-color': '#6495ED', 'color': '#F8F8FF'});",
                                        "}")))
)
output$vision_tabla_densidad <- renderDataTable(
  datatable(vision_densidad, rownames = FALSE, options = list(dom = 'ft', orderClasses = TRUE,
                                      language = list(paginate = list('next' = "Siguiente", previous = "Anterior"),
                                                      search = "Buscar:", zeroRecords = "Sin resultados de búsqueda"),
                                      initComplete = JS(
                                        "function(settings, json) {",
                                        "$(this.api().table().header()).css({'background-color': '#6495ED', 'color': '#F8F8FF'});",
                                        "}")))
)
output$vision_tabla_ODS <- renderDataTable(
  datatable(vision_ODS, rownames = FALSE, options = list(dom = 'ft', orderClasses = TRUE,
                                       language = list(paginate = list('next' = "Siguiente", previous = "Anterior"),
                                                       search = "Buscar:", zeroRecords = "Sin resultados de búsqueda"),
                                       initComplete = JS(
                                         "function(settings, json) {",
                                         "$(this.api().table().header()).css({'background-color': '#6495ED', 'color': '#F8F8FF'});",
                                         "}")))
)
```

```
}

# Run the application
shinyApp(ui = ui, server = server)
```