Modelos 3PRF - Ministerio de Hacienda y Crédito Público (DGPM)

Automatización del modelo 3PRF

Ministerio de Hacienda y Crédito Público - DGPM

2024-04-26

El presenta manual es una descripción detallada y completa de todo el proceso que se requiere para ejecutar el modelo 3PRF del ministerio de Hacienda en R, desde la descarga y obtención de los datos hasta la estimación del modelo. Para ello, se explicará en sucintamente las tecnologías necesarias para correr el modelo, los paquetes requeridos, las funciones construidas y en general el código necesario para poder correr el modelo. La idea es que el documento sirva como guía para comprender y familiarizarse con todo el proceso requerido para hacer estimaciones con el modelo 3PRF en R, y que también sirva como guía en caso de que haya que hacer mantentivmiento o modificaciones al código. El documento contiene incialmente una sección preliminar donde se dan unas indicaciones generales del docoumento, luego una introducción donde se presenta de forma breve el modelo, posteriormente se describen en detalle cada una de las etapas requeridas para correr el modelo las cuales son: 1) la descarga y obtención de los datos, 2) el procesamiento de las base de datos, 3) la desestacionalización de las variables del modelo y 4) la estimación del modelo, finalmente se encuentran los apéndices al documento.

Tabla de contenidos

1 Preliminares

Siglas

- **Metodologías y modelos:**
  - **3PRF** Filtro de Regresión en Tres Pasos - Three-Pass-Regression-Filter (por sus siglas en inglés)
- **Indicadores económicos:**
  - **ISE** Índice de Seguimiento a la Economía
  - **PIB** Producto Interno Bruto
- **Instituciones:**
  - **MHCP** Ministerio de Hacienda y Crédito Público
  - **Minenergía** Ministerio de Minas y Energía
  - **DANE** Departamento Administrativo Nacional de Estadística
  - **DIAN** Dirección de Impuestos y Aduanas Nacionales
  - **FNC** Federación Nacional de Cafeteros
  - **Aerocivil** Aeronautica Civil
  - **Banrep** Banco de la República
  - **Camacol** Cámara Colombiana de la Construcción

Conocimientos técnicos útiles que aprenderá en el documento

- Tecnologias web y web scraping mediante las librerias `rvest` en `R` y `selenium` en `Python`
- Procesamiento avanzado de datos en Excel mediante las librerías `tidyxl` y `unpivotr`
- Manejo de *expresiones regulares* mediante la librería `stringr`
- Manejo de excepciones mediante mediante la sentencia `try and catch` de R
- Programación funcional básica en R
- Desestacionalización de series de tiempo en R
- Estimación de un Filtro de Regresión en Tres Pasos en R
- Manejo de series de tiempo en R mediante librerias especializadas como lo son `xts` y `fable`

2 Introducción

En la actualidad, existen difrentes indicadores macroeconómicos que analizados de manera conjunta pueden dar un diagnóstico de la situación económica del país, lo que resulta muy útil para la toma de decisiones de los agentes y por supuesto para la formulación de política pública por parte entidades como el Ministerio de Hacienda y Crédito Público. Dichos indicadores, pueden ser encontrados en las diferentes páginas

webs de diversos tipos de organizaciones y agencias como lo son la entidades gubernamentales, las agencias de estadísticas, los gremios y en general el sector privado. El objetivo del presente documento, consiste primoridalmente en explicar de manera detallada y … el proceso completo de estimación del modelo de Filtro

2.1 Descripción general del proceso completo necesario para estimar el modelo 3PRF

El proceso completo necesario para estimar el modelo ***3PRF*** consiste en**:**

1. Descarga y obtención de los datos
   - Descarga automática
     - Vía `rvest en R`
     - Vía `selenium en python`
   - Obtención manual
2. Procesamiento de las bases de datos para introducir la información de manera automática al modelo
3. Desestacionalización de las series de tiempo que entran dentro del modelo **3PRF**
4. Estimación del modelo **3PRF** en R

A continuación, se hará una descripción más detallada de cada una de las etapas necesarias para el proceso de estimación del modelo 3PRF.

2.2 Manejo de directorios y bases de datos

3 Descarga y obtención de los datos para el modelo 3PRF

3.1 Tecnologías necesarias para realizar Web scraping

3.1.1 HTML

Es un lenguaje de tipo declarativo que se utiliza virtualmente en toda página que se encuentre en internet, es decir, necesariamente toda página web tiene detrás un archivo HTML que le da su estructura y que es leído por el navegador para poder renderizar la página para el ususario. Como lo indica la 2023 StackOverflow Developer Survey el 52.9% de los desarrolladores que respondieron dicha encuesta (tal vez la encuesta más representativa a nivel mundial en términos de tendencias de programación) utiliza `HTML/CSS`, lo que lo hace que estos dos lenguajes de programación sean los segundos lenguajes más utilizados del mundo, al menos según las métricas de esta encuesta.

3.1.2 CSS

3.1.3 JavaScript

`JavaScript`, a menudo abreviado como `JS`, es un lenguaje de programación y tecnología central de la Web, junto con `HTML` y `CSS`. La mayor parte de los sitios web hoy en día utilizan `JavaScript` en el lado del cliente (`Front-end`) para el comportamiento de la página web. Como lo indica la 2023 StackOverflow Developer Survey el 65.61% de los desarrolladores que respondieron dicha encuesta (tal vez la encuesta más representativa a nivel mundial en términos de tendencias de programación) utiliza `JavaScript`, lo que lo hace el lenguaje de programación más popular y utilizado del mundo, al menos según las métricas de esta encuesta. JavaScript fue diseñado por Brendan Eich, el mismo desarrollador de los navegadores de código abierto Mozilla Firefox y Brave.

3.2 `Web scraping`?

3.3

¿Qué es el `Web scraping`?

Es un conjunto de técnicas que permite descargar información de la web de manera automática. A nivel práctico, el `web scraping` consiste en la realización de un programa capaz de acceder a sitios web y recuperar información dentro de éstos. Para ello, el programa mediante una librería especializada accede al archivo `HTML` de la página web de la cuál se quiere obtener información e interactua con dicho archivo `HTML` para obtener los datos requeridos.

Detrás de dicho proceso, cuándo una libería interactúa con una página web, lo que está haciendo es enviar y recibir solicitudes `HTTP` que es la forma en la que los diferentes servidores que conforman la web se comunican, es decir, cada solicitud que se haga desde una librería, como lo puede ser por ejemplo descargar una base de datos de una página web, lo que está haciendo es enviar una solicitud `HTTP` al servidor que contiene la página web para que se haga la descarga de la base.

El `web scraping` es un procedimiento muy conveniente para automatizar tareas repetitivas, como lo puede ser la descarga de datos, o cuándo haya que descargar un volumen muy grande de información de la web y sea incoveniente realizar dicha tarea a mano. Por ello, las técnicas de web scraping son muy convenientes para automatizar procesos dentro de la DGPM del ministerio, y en particular para automatizar la obtención de los datos requeridos para la estimación del modelo 3PRF.

3.4 ¿Cómo realizar Web scraping?

3.4.0.1 Web scraping con páginas estáticas

Una página se puede considerar *estática* cuando no requiere el uso activo de un navegador que corra o ejectue `JavaScript` detrás para poder renderizar la página, es decir, con solo el archivo HTML es posible renderizar completamente la página web.

3.4.0.2 Web scraping con páginas dinámicas

Una página se puede considerar *dinámica* cuando requiere el uso activo de un navegador que corra o ejectue `JavaScript` detrás para poder renderizar la página, es decir, el navegador activamente utiliza `JavaScript` para que interactúe con los diferentes nodos o `tags` del archivo `HTML` y de esta forma pueda darle el dinamismo en tiempo real a la página web.

3.5 Tablas de variables del modelo 3PRF

Actualmente, el **modelo 3PRF** emplea en su estimación un total de 72 variables provenientes de diferentes fuentes de datos como lo son instituciones gubernamentales, agencias de estadísticas, gremios y diversas entidades privadas.

⚠ Las variables para el **modelo 3PRF** provienen de las siguientes fuentes de información:

- **Dane**: 26 variables en total
- **Banrep**: 13 variables en total

3.5.1 Tabla con las variables que actualmente son descargadas y procesadas de manera automática

De las 72 variables, 46 de éstas actualmente son descargadas y procesadas de manera automática.

Tabla 1: Tabla de las variables del modelo 3PRF cuya obtención ya fue automatizada vía *`web scraping`*

No.	Nombre de la variable	Fuente	Modo de obtención
1	ISE	DANE	Automática - Web Scraping (rvest en R)
2	Índice de producción real de la industria	DANE	Automática - Web Scraping (rvest en R)
3	Índice de ventas reales del comercio	DANE	Automática - Web Scraping (rvest en R)
4	Importaciones en dólares	DANE	Automática - Web Scraping (rvest en R)
5	Importaciones de bienes de consumo	DANE	Automática - Web Scraping (rvest en R)
6	Importaciones de bienes intermedios	DANE	Automática - Web Scraping (rvest en R)
7	Importaciones de bienes de capital	DANE	Automática - Web Scraping (rvest en R)
8	Exportaciones en dólares	DANE	Automática - Web Scraping (rvest en R)
9	Exportaciones de bienes tradicionales	DANE	Automática - Web Scraping (rvest en R)
10	Exportaciones de bienes no tradicionales	DANE	Automática - Web Scraping (rvest en R)
11	Despachos de cemento	DANE	Automática - Web Scraping (rvest en R)
12	Área licenciada de construcción	DANE	Automática - Web Scraping (rvest en R)
13	IPP	DANE	Automática - Web Scraping (rvest en R)
14	IPC	DANE	Automática - Web Scraping (rvest en R)
15	Ingresos reales de los hoteles	DANE	Automática - Web Scraping (rvest en R)
16	Tasa de desempleo	DANE	Automática - Web Scraping (rvest en R)
17	Tasa de ocupación	DANE	Automática - Web Scraping (rvest en R)
18	Ocupados	DANE	Automática - Web Scraping (rvest en R)
19	Desocupados	DANE	Automática - Web Scraping (rvest en R)
20	Peso en pie de sacrificio de ganado vacuno	DANE	Automática - Web Scraping (rvest en R)
21	Peso en pie de sacrificio de ganado porcino	DANE	Automática - Web Scraping (rvest en R)
22	Producción de carbón	DANE	Automática - Web Scraping (rvest en R)
23	Exportaciones de carbón	DANE	Automática - Web Scraping (rvest en R)
24	Índice de empleo de la industria manufacturera	DANE	Automática - Web Scraping (rvest en R)
25	Índice de personal ocupado del comercio minorista	DANE	Automática - Web Scraping (rvest en R)
26	IPC Core	Banrep	Automática - Web Scraping (selenium en python)

No.	Nombre de la variable	Fuente	Modo de obtención
27	IPC Alimentos	Banrep	Automática - Web Scraping (selenium en python)
28	IPC Regulados	Banrep	Automática - Web Scraping (selenium en python)
29	Tasa de cambio real	Banrep	Automática - Web Scraping (selenium en python)
30	Tasa de cambio nominal	Banrep	Automática - Web Scraping (selenium en python)
31	Curva cero cupón a 1 año (COP)	Banrep	Automática - Web Scraping (selenium en python)
32	Curva cero cupón a 5 años (COP)	Banrep	Automática - Web Scraping (selenium en python)
33	Curva cero cupón a 10 años (COP)	Banrep	Automática - Web Scraping (selenium en python)
34	Cartera consumo	Banrep	Automática - Web Scraping (selenium en python)
35	Cartera hipotecaria	Banrep	Automática - Web Scraping (selenium en python)
36	Cartera comercial	Banrep	Automática - Web Scraping (selenium en python)
37	Remesas en dólares	Banrep	Automática - Web Scraping (selenium en python)
38	Lanzamientos de vivienda	Camacol	Automática - Web Scraping (rvest en R)
39	Iniciaciones de vivienda	Camacol	Automática - Web Scraping (rvest en R)
40	Ventas de vivienda	Camacol	Automática - Web Scraping (rvest en R)
41	Recaudo de IVA interno	DIAN	Automática - Web Scraping (rvest en R)
42	Recaudo de impuestos externos	DIAN	Automática - Web Scraping (rvest en R)
43	Transporte de pasajeros aéreos	Aerocivil	Automática - Web Scraping (rvest en R)
44	Transporte aéreo de carga y correo	Aerocivil	Automática - Web Scraping (rvest en R)
45	Producción de café	FNC	Automática - Web Scraping (rvest en R)
46	Precio internacional del café	FNC	Automática - Web Scraping (rvest en R)

3.5.2 Tabla con las variables que actualmente se obtienen de manera manual

De las 72 variables del **modelo 3PRF**, 26 de éstas actualmente se obtienen de manera manual.

Tabla 2: Tabla de las variables del modelo 3PRF cuya obtención sigue siendo manual

No.	Nombre de la variable	Fuente	Modo de obtención
1	Gastos en personal del gobierno	MHCP	Variable interna MinHacienda
2	Gastos en bienes del gobierno	MHCP	Variable interna MinHacienda
3	Exportaciones de flores	MHCP	Variable interna MinHacienda
4	Exportaciones de banano	MHCP	Variable interna MinHacienda
5	Precio internacinoal del petróleo Brent	Bloomberg	Subfiscal - Monitor Fiscal
6	CDS a 5 años	Bloomberg	Pasante de externo Ticket: COLOM CDS USD SR 5Y D14 Corp
7	Tasa de usura	Bloomberg	Pasante de externo Ticket: CUSURCC Index
8	Tasa de interés FED	Bloomberg	Pasante de externo Ticket: FDTR Index
9	Despachos de combustible Gasolina	Minenergía	Variable interna MinHacienda (Subdirección fiscal)
10	Despachos de combustible ACPM	Minenergía	Variable interna MinHacienda (Subdirección fiscal)
11	Producción de petróleo	Minenergía	Variable interna MinHacienda (Informe mensual de petróleo)
12	Índice de Confianza del Consumidor	Fedesarrollo	Variable interna MinHacienda (Macro - EOC)
13	Índice de Confianza Industrial	Fedesarrollo	Variable interna MinHacienda (Macro - EOE)
14	Índice de Confianza Comercial	Fedesarrollo	Variable interna MinHacienda (Macro - EOE)
15	Confianza de los hogares	Davivienda	Descarga manual
16	PMI Davivienda	Davivienda	Descarga manual
17	Ventas de carros nuevos	Andemos	Descarga manual
18	Ventas de motos nuevas	Andemos	Descarga manual
19	Demanda de energía	XM	Variable interna MinHacienda (Ind. Actividad Real)
20	Demanda de energía no regulada	XM	Variable interna MinHacienda (Ind. Actividad Real)
21	Transporte de carga en toneladas	MinTransporte	Descarga manual
22	Indicadores del ciclo ENOS	ANOA	Variable interna MinHacienda (IDEAM)
23	Producción de palma africana	Fedepalma	Descarga manual
24	Producción de pollo	Fenavi	Descarga manual
25	Horas efectivas trabajadas	DANE	Pasante de laboral
26	Tasa de interés del Banrep	Banrep	Pasante de externo Ticket: CORRRMIN Index

⚠ Resumen de la obtención de variables del 3PRF

3.5.3 Descarga y procesamiento automático

Actualmente, como muestra la tabla Tabla 1, 46 de las 72 variables del 3PRF pueden ser descargadas vía `web scraping`. De las 46 variables que se pueden obtener vía `web scraping`, 34 de ellas se obtienen mediante el uso del paquete **`rvest`** en `R`, y el resto se obtiene

mediante el uso del paquete **selenium** en `python`.
La descarga de las 46 variables de manera automática vía `web scraping`, se obtienen así:

- **DANE** : 25 variables vía `web scraping` (`rvest en R`)
- **Banrep:** 12 variables vía `web scraping` (`selenium en python`)
- **Camacol**: 3 variables vía `web scraping` (`rvest en R`)
- **DIAN**: 2 variables vía `web scraping` (`rvest en R`)
- **Aerocivil**: 2 variables vía `web scraping` (`rvest en R`)
- **FNC**: 2 variables vía `web scraping` (`rvest en R`)

3.5.4 Procesamiento manual

Las otras 26 variables que requiere el modelo se deben actualizar de manera manual como lo muestra la tabla Tabla 2.

3.6 Base de datos necesaria para la descarga de los datos vía Web scraping

Dentro del directorio de trabajo `input_web_scraping` se encuentra la base de datos `input_web_scraping` cuya pestaña `descarga` contiene la base de datos necesaria para realizar el proceso de web scraping en R de la bases de datos. El contenido de dicha base de datos se encuentra en la Tabla 3.

3.7 Paquetes para realizar el Web scraping del modelo 3PRF

3.7.1 Paquetes para realizar Web Scraping en R

3.7.1.1 rvest

El paquete principal para la descarga de bases de datos vía **web scraping** en R es `rvest`.

! rvest

En palabras del autor, `rvest` le ayuda a extraer (o recolectar) datos de páginas web. Inspirado en bibliotecas como `Beautiful Soup` y `RoboBrowser`.
Si está realizando varias solicitudes de descarga vía **web scraping** en la misma página web, se recomienda utilizar `rvest` junto con el paquete `polite`. El paquete `polite` garantiza que se respete el archivo `robots.txt` de la página y no sobrecarga el sitio con demasiadas solicitudes(Wickham 2024).

3.7.1.2 polite

3.7.2 Paquetes para realizar Web Scraping en python

3.7.2.1 selenium

3.8 Código para realizar el Web scraping del modelo 3PRF

3.8.1 Paquetes neceasrios para realizar el Web scraping del modelo 3PRF

Se importan los paquetes necesarios para ejectuar las funciones dentro del código.

```
### Paquetes

# Paquete multiproposito de R
# (De lejos el paquete más importante de todo R.
# Fundamental para hacer "análisis de datos" en R!!)
library(tidyverse)

# Paquetes adicionales del Tidyverse
library(lubridate) # Para el manejo de fechas en R
```

Tabla 3: Base de datos necesaria para la descarga de los datos vía Web scraping

(a) Variables: global_url, tipo_selector, Fuente, nombre_base_descarga

global_url	tipo_selector	Fuente	nombre_base_descarga
https://www.dane.gov.co	css_selector	DANE	DANE_ISE_9_actividades.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_ISE_12_actividades.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_EMMET_territorial.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_EMC.xlsx
https://camacol.co	xpath	CAMACOL	CAMACOL_Tablas_coyuntura.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_Importaciones.xls
https://www.dane.gov.co	css_selector	DANE	DANE_Exportaciones.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_ECG.xlsx
https://www.aerocivil.gov.co	xpath	AEROCIVIL	AEROCIVIL_Estadisticas.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_ELIC.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_IPP.xlsx
https://federaciondecafeteros.org	xpath	FNC	FNC_Estadisticas.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_IPC.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_EMA.xlsx
https://www.dane.gov.co	css_selector	DANE	DANE_Mercado_laboral.xlsx
https://www.dian.gov.co	css_selector	DIAN	Recaudo_impuestos_DIAN_colombia
https://www.dane.gov.co	css_selector	DANE	DANE_Sacrificio_ganado.xls
https://www.dane.gov.co	css_selector	DANE	DANE_IPI.xlsx

(b) Variables: enlace_variable_3PRF

enlace_variables_3PRF
https://www.dane.gov.co/index.php/estadisticas-por-tema/cuentas-nacionales/indicador-de-seguimiento-a-la-economia-ise
https://www.dane.gov.co/index.php/estadisticas-por-tema/cuentas-nacionales/indicador-de-seguimiento-a-la-economia-ise
https://www.dane.gov.co/index.php/estadisticas-por-tema/industria/encuesta-mensual-manufacturera-con-enfoque-territorial-emmet
https://www.dane.gov.co/index.php/estadisticas-por-tema/comercio-interno/encuesta-mensual-de-comercio-emc
https://camacol.co/nuestro-sector/informacion-economica
https://www.dane.gov.co/index.php/estadisticas-por-tema/comercio-internacional/importaciones
https://www.dane.gov.co/index.php/estadisticas-por-tema/comercio-internacional/exportaciones
https://www.dane.gov.co/index.php/estadisticas-por-tema/construccion/estadisticas-de-cemento-gris
https://www.aerocivil.gov.co/atencion/estadisticas-de-las-actividades-aeronauticas/bases-de-datos
https://www.dane.gov.co/index.php/estadisticas-por-tema/construccion/licencias-de-construccion
https://www.dane.gov.co/index.php/estadisticas-por-tema/precios-y-costos/indice-de-precios-del-productor-ipp
https://federaciondecafeteros.org/wp/estadisticas-cafeteras/
https://www.dane.gov.co/index.php/estadisticas-por-tema/precios-y-costos/indice-de-precios-al-consumidor-ipc
https://www.dane.gov.co/index.php/estadisticas-por-tema/servicios/encuesta-mensual-de-alojamiento-ema
https://www.dane.gov.co/index.php/estadisticas-por-tema/mercado-laboral/empleo-y-desempleo
https://www.dian.gov.co/dian/cifras/Paginas/EstadisticasRecaudo.aspx
https://www.dane.gov.co/index.php/estadisticas-por-tema/agropecuario/encuesta-de-sacrificio-de-ganado
https://www.dane.gov.co/index.php/estadisticas-por-tema/industria/indice-de-produccion-industrial-ipi

(c) Variables: tag_selector_descarga

tag_selector_descarga
#economia > section > div.info-tecnica > div.docs-tecnicos > table > tbody > tr:nth-child(2) > td:nth-child(5) > a
#economia > section > div.info-tecnica > div.docs-tecnicos > table > tbody > tr:nth-child(3) > td:nth-child(5) > a
#cnpv > div > div > table:nth-child(2) > tbody > tr > td > table > tbody > tr:nth-child(3) > td.v-align > a
#cnpv > div > div > table:nth-child(2) > tbody > tr > td > table:nth-child(10) > tbody > tr:nth-child(3) > td.v-align > a
//*(**id?**, =“block-descargabletablasdecoyuntura”)/div/div/div/p[2]/a
#t3-content > div > div.com-content-article___body > table:nth-child(1) > tbody > tr > td > table:nth-child(6) > tbody > tr > td:nth-child(
#t3-content > div > div.com-content-article___body > table.table.table-hover.wideLink > tbody > tr:nth-child(4) > td > p > strong > a
#t3-content > div > div.com-content-article___body > table > tbody > tr > td > table > tbody > tr > td:nth-child(2) > div > a
//*(**id?**)/ul
#t3-content > div > div.com-content-article___body > table.table.table-hover.wideLink > tbody > tr:nth-child(2) > td > p > strong > a
#t3-content > div > div.com-content-article___body > table.table.table-hover.wideLink > tbody > tr:nth-child(2) > td > p > strong > a
//*(**id?**)/div/div/div/div[2]/section[10]/div/div[1]/div/div/div/div/div[2]/div/a
#t3-content > div > div > div.row > div:nth-child(3) > p:nth-child(2) > a
#t3-content > div > div.com-content-article___body > table:nth-child(1) > tbody > tr > td > table:nth-child(15) > tbody > tr > td > div >
#rlta-panel-empleo-y-desocupacion > div > div > div.txt_info-tecnica > div.docs-tecnicos > table > tbody > tr:nth-child(5) > td:nth-child(5)
#WebPartWPQ6 > div.ms-rtestate-field > ul > li > p > a
#rlta-panel-resultados-historicos > div > table > tbody > tr:nth-child(1) > td > p > strong > a
#t3-content > div > div.com-content-article___body > table:nth-child(1) > tbody > tr > td > table > tbody > tr > td:nth-child(2) > div > a

```r
library(stringr) # Para el manejo de "Strings" en R

# Paquetes para realizar "Web scraping" en R

## Paquetes para trabajar con páginas web que no necesiten Javascript
library(rvest)
library(xml2)

## Paquetes para trabajar con páginas web que necesiten Javascript
library(RSelenium)

## Paquetes adicionales que falicitan el "Web Scraping" en R
library(httr2) # Paquete para enviar HTTP request a "servidores web"
library(polite) # Paquete para hacer uso responsable del "Web Scraping"

# Paquetes estandándares para manejo de archivos de Excel (xlsx) en R
library(readxl)
library(openxlsx)
library(data.table)
library(writexl)

# Paquetes para lidiar con "archivos complejos" de Excel
# que tengan información no rectangular
library(tidyxl) # Importar datos de Excel sin forzarlos a formar un rectángulo
library(unpivotr) # Para lidiar con información no tabular de Excel

# Paquete para manejo avanzado de series de tiempo en R
library(xts)
```

i Descripción de los paquetes que requiere el Script

- Paquetes de *propósito general*:
  - **`tidyverse`:** Paquete multiproposito de `R` que permite realizar varias tareas, entre ellas, analizar, manipular y gráficar datos.
    - Los siguientes paquetes hacen parte del `tidyverse`:
      1. `dplyr`: Paquete estándar para en análisis y manipulación de bases de datos.
      2. `ggplot2`: Paquete estándar para graficación en `R`.
      3. `tidyr`: Paquete para manipulaciones avanzadas de bases de datos.
      4. `forcats`: Paquete para el manejo de datos categóricos en `R`.
      5. `stringr`: Paquete para el manejo de *strings* (cadenas de carácteres en `R`). Permite entre otras cosas, el manejor de `regex`, para buscar y manipular patrones dentro de cadenas de texto.
      6. `readr`: Importar archivos de carácters csv y otros parecidos a R.
      7. `purrr`: Para realizar *programación funcional* en `R`.
      8. `tibble`: Para manipular objetos de tipo tibble, que son dataframes en `R` con funcionalidades adicionales.
    - Informacion adicional:
      - Vídeo de Posit sobre prácticas adecuadas en el tidyverse
- Paquetes adicionales del *tidyverse*:
  - lubridate: Paquete para el manejo de fechas y días en R.
- Paquetes para realizar **web scraping** en págianas web *no* necesiten Javascript
  - rvest: Paquete principal para realizar web scraping en R. Hace también parte del universo de conjuntos del *tidyverse.*
  - xml2: Paquete diseñado para trabajar archivos HTML y XML en R
- Paquetes para realizar **web scraping** en págianas web necesiten Javascript
  - RSelenium: Provee una interfaz para el uso de **Selenium** en R.
- Paquetes adicionales que falicitan el **web scraping** en R
  - httr2: Paquete para enviar **HTTP** request a servidores web. Hace también parte del universo de conjuntos del *tidyverse.*
  - polite: Paquete para hacer uso responsable del web scraping
- Paquetes estandándares para manejo de archivos de **Excel (.xlsx)** en R

  - readxl: Paquete para abrir archivos excel (.xlsx) en R. Hace también parte del universo de conjuntos del *tidyverse.*
  - openxlsx: Paquete alternativo para archivos excel (.xlsx) en R
  - data.table: Paquete que provee funcionalidades adicionales para el análisis y procesamiento de *dataframes* en R.
  - writexl: Paquete para exportar dataframes o lista de dataframes de R a Excel.
- Paquetes para manipular y procesar "archivos complejos" de **Excel** que tengan información no rectangular
  - tidyxl: Importar datos de Excel sin forzarlos a tener un formato rectangular dentro de R.
  - unpivotr: Para lidiar con información no tabular de Excel en R. Contiene un conjunto de funciones extremadamente útil para procesar archivos Excel complejos.
- Paquete para manejo avanzado de series de tiempo en R
  - xts: Paquete que permite realizar un manejo avanzado de series de tiempo y facilitar el análisis con éstas. Permite trabajar series de tiempo con frecuencias regulares (e.g. mensuales, trimestrales) y no regulares, como lo pueden ser datos intradiarios que no necesariamente tiene una distancia temporal prestablecida. Muy utilizado en análisis con series de tiempo financieras.

3.8.2 Funciones para el proceso de desacarga de datos vía Web scraping de las diferentes bases

3.8.2.1 Función para la descarga de datos del DANE

```
# Función para hacer web scraping en el DANE
descarga_DANE = function(global_url,
                         enlace_variable_3PRF,
                         tag_selector_descarga,
                         tipo_selector,
                         nombre_base){

  # Archivo HTML de la página web importada a R
  html = read_html(enlace_variable_3PRF)

  # La selección del tag cambia dependiente de si se uso el "xpath" o el
  # "CSS selector" del tag para identificarlo dentro del archivo HTML
  if(tipo_selector == "xpath"){

    # URL del archivo Excel para descargar
    excel_url = html %>%
      html_elements(xpath = tag_selector_descarga) %>%
      html_attr("href")

  }else if(tipo_selector == "css_selector"){

    # URL del archivo Excel para descargar
    excel_url = html %>%
      html_elements(tag_selector_descarga) %>%
      html_attr("href")
  }

  # Descarga del archivo
  download.file(paste0(global_url, excel_url), destfile = nombre_base,
                mode = "wb")
}
```

3.8.2.2 Función para la descarga de datos de Camacol

```
# Función para hacer web scraping en Camacol
descarga_Camacol = function(global_url,
                            enlace_variable_3PRF,
                            tag_selector_descarga,
                            tipo_selector,
                            nombre_base){

  # Archivo HTML de la página web importada a R
```

```
  html = read_html(enlace_variable_3PRF)

  # La selección del tag cambia dependiente de si se uso el "xpath" o el
  # "CSS selector" del tag para identificarlo dentro del archivo HTML
  if(tipo_selector == "xpath"){

    # URL del archivo Excel para descargar
    excel_url = html %>%
      html_elements(xpath = tag_selector_descarga) %>%
      html_attr("href")

  }else if(tipo_selector == "css_selector"){

    # URL del archivo Excel para descargar
    excel_url = html %>%
      html_elements(tag_selector_descarga) %>%
      html_attr("href")
  }

  # Descargar del archivo
  download.file(paste0(global_url, excel_url), destfile = nombre_base,
                mode = "wb")

}
```

3.8.2.3 Función para la descarga de datos de la Aerocivil

```
# Función para hacer web scraping en la Aerocivil
descarga_Aerocivil = function(enlace_variable_3PRF,
                              tag_selector_descarga,
                              tipo_selector,
                              nombre_base){

  # Archivo HTML de la página web importada a R
  html = read_html(enlace_variable_3PRF)

  # La selección del tag cambia dependiente de si se uso el "xpath" o el
  # "CSS selector" del tag para identificarlo dentro del archivo HTML
  if(tipo_selector == "xpath"){

    # URL del archivo para descargar
    excel_url = html %>%
      html_elements(xpath = tag_selector_descarga) %>%
      html_elements("li:first-child") %>%
      html_elements("div:first-child") %>%
      html_elements(xpath = "./a[1]") %>%
      html_attr("href")

  }

  # Para que no tenga problemas con la base de datos
  encoded_url <- URLencode(excel_url)

  # Descargar del archivo
  download.file(encoded_url, destfile = nombre_base, mode = "wb")

}
```

3.8.2.4 Función para la descarga de datos de la FNC

```
# Función para hacer web scraping en la Federación Nacionl de Cafeteros
descarga_FNC = function(enlace_variable_3PRF,
                        tag_selector_descarga,
                        tipo_selector,
```

```
                        nombre_base){

  # Archivo HTML de la página web importada a R
  html = read_html(enlace_variable_3PRF)

  # La selección del tag cambia dependiente de si se uso el "xpath" o el
  # "CSS selector" del tag para identificarlo dentro del archivo HTML
  if(tipo_selector == "xpath"){

    # URL del archivo Excel para descargar
    excel_url = html %>%
      html_elements(xpath = tag_selector_descarga) %>%
      html_attr("href")

  }else if(tipo_selector == "css_selector"){

    # URL del archivo Excel para descargar
    excel_url = html %>%
      html_elements(tag_selector_descarga) %>%
      html_attr("href")
  }


  # Descarga del archivo
  download.file(excel_url, destfile = nombre_base, mode = "wb")
}
```

3.8.2.5 Función para la descarga de datos de la DIAN

```
# Función para hacer web scraping en la DIAN
descarga_DIAN = function(global_url,
                         enlace_variable_3PRF,
                         tag_selector_descarga,
                         tipo_selector,
                         nombre_base){

  # Archivo HTML de la página web importada a R
  html = read_html(enlace_variable_3PRF)

  # La selección del tag cambia dependiente de si se uso el "xpath" o el
  # "CSS selector" del tag para identificarlo dentro del archivo HTML
  if(tipo_selector == "xpath"){

    # URL del archivo Excel para descargar
    excel_url = html %>%
      html_elements(xpath = tag_selector_descarga) %>%
      html_attr("href")

  }else if(tipo_selector == "css_selector"){

    # URL del archivo Excel para descargar
    excel_url = html %>%
      html_elements(tag_selector_descarga) %>%
      html_attr("href")
  }

  # Creción del archivo temporal para almacenar el archivo .zip
  temp <- tempfile()

  # Descarga del archivo ".zip"
  download.file(paste0(global_url, excel_url), temp, mode = "wb")

  # Descompresión del archivo ".zip". Se obtiene el archivo ".xlsx"
  unzip(zipfile = temp, exdir = "./")
```

```
  # Borra el archivo temporal del sistema
  unlink(temp)
}
```

3.8.3 Función para la descarga de todos los datos vía la base da datos `input_web_scraping`

```
# Función para descarga de todos los datos
descarga_datos_3PRF = function(input_3PRF_descarga, tmp_sleep){

  for (i in 1:nrow(input_3PRF_descarga)){

    if(input_3PRF_descarga[i,]$Fuente == "DANE"){

      # Función para descargar bases del DANE
      descarga_DANE(global_url = input_3PRF_descarga[i,]$global_url,
                    enlace_variable_3PRF = input_3PRF_descarga[i,]$enlace_variables_3PRF,
                    tag_selector_descarga = input_3PRF_descarga[i,]$tag_selector_descarga,
                    tipo_selector = input_3PRF_descarga[i,]$tipo_selector,
                    nombre_base = input_3PRF_descarga[i,]$nombre_base_descarga)

      # Se pausa el programa "tmp_sleep" segudnos para que no haya problemas en la descarga
      # de los archivos
      Sys.sleep(tmp_sleep)

    }else if (input_3PRF_descarga[i,]$Fuente == "CAMACOL"){

      # Función para descargar bases de Camacol
      descarga_Camacol(global_url = input_3PRF_descarga[i,]$global_url,
                       enlace_variable_3PRF = input_3PRF_descarga[i,]$enlace_variables_3PRF,
                       tag_selector_descarga = input_3PRF_descarga[i,]$tag_selector_descarga,
                       tipo_selector = input_3PRF_descarga[i,]$tipo_selector,
                       nombre_base = input_3PRF_descarga[i,]$nombre_base_descarga)

      # Se pausa el programa "tmp_sleep" segudnos para que no haya problemas en la descarga
      # de los archivos
      Sys.sleep(tmp_sleep)

    }else if(input_3PRF_descarga[i,]$Fuente == "AEROCIVIL"){

      # Función para descargar bases Aerocivil
      descarga_Aerocivil(enlace_variable_3PRF = input_3PRF_descarga[i,]$enlace_variables_3PRF,
                         tag_selector_descarga = input_3PRF_descarga[i,]$tag_selector_descarga,
                         tipo_selector = input_3PRF_descarga[i,]$tipo_selector,
                         nombre_base = input_3PRF_descarga[i,]$nombre_base_descarga)

      # Se pausa el programa "tmp_sleep" segudnos para que no haya problemas en la descarga
      # de los archivos
      Sys.sleep(tmp_sleep)

    }else if(input_3PRF_descarga[i,]$Fuente == "FNC"){

      # Función para descargar bases FNC
      descarga_FNC(enlace_variable_3PRF = input_3PRF_descarga[i,]$enlace_variables_3PRF,
                   tag_selector_descarga = input_3PRF_descarga[i,]$tag_selector_descarga,
                   tipo_selector = input_3PRF_descarga[i,]$tipo_selector,
                   nombre_base = input_3PRF_descarga[i,]$nombre_base_descarga)

      # Se pausa el programa "tmp_sleep" segudnos para que no haya problemas en la descarga
      # de los archivos
      Sys.sleep(tmp_sleep)

    }else if(input_3PRF_descarga[i,]$Fuente == "DIAN"){

      # Función para descargar bases DIAN
```

```
      descarga_DIAN(global_url = input_3PRF_descarga[i,]$global_url,
                    enlace_variable_3PRF = input_3PRF_descarga[i,]$enlace_variables_3PRF,
                    tag_selector_descarga = input_3PRF_descarga[i,]$tag_selector_descarga,
                    tipo_selector = input_3PRF_descarga[i,]$tipo_selector,
                    nombre_base = input_3PRF_descarga[i,]$nombre_base_descarga)

      # Se pausa el programa "tmp_sleep" segudnos para que no haya problemas en la descarga
      # de los archivos
      Sys.sleep(tmp_sleep)

    }

  }

}
```

4 Procesamiento de las base de datos para introducir información de manera automática al modelo

4.1 Tecnologías necesarias para realizar el procesamiento automático de las bases de datos del 3PRF

4.1.1 Tidy Data

Hoy en día, entidades como Posit (compañía de RStudio y el paquete `tidyverse`) y autores como Hadley Wickham (principal desarrollador y supervisor del paquete tidyverse), recomiendan que los datos se encuentren en formato ***tidy***. Es así, que hoy en la *ciencia de datos* el estándar para el manejo y procesamiento de los datos requiera que en su mayoría los datos se encuentren en dicho formato ***tidy***.

En su artículo Tidy Data, Hadley Wickham argumenta que para que una base de datos se pueda considerar que está en formato `tidy` se tiene que satisfacer las siguientes 3 caracterísicas dentro de la base (Wickham 2014):

1. Cada variable tiene asociada una columna
2. Cada observación forma una fila
3. Cada tipo de unidad observacional forma una columna

Los conjuntos de datos reales generalmente violan dichos tres principios básicos que definen que una base de datos se encuentre en formato `tidy`, por lo que en muchos casos es necesario transformar la base de datos a este tipo de formato. Las cinco razones principales por las que una base de datos no se encuentra en formato `tidy` son (Wickham 2014):

- Los encabezados de columna son los valores valores en lugar del nombre de una variable
- Múltiples variables se encuentran almacenadas en una sola columna
- Las variables se encuentran tanto almacenadas en filas como en columnas
- Múltiples tipos de unidades observacionales se encuentran almacenadas en la misma tabla
- Una unidad observacional se encuentra almacenada en múltiples tablas

En caso de que la base de datos presente alguna de las condiciones mencioandas arriba, se hace necesario procesar la base de datos de tal forma que esta pase a un formato `tidy` que facilite el análisis y procesamiento de los datos. Para transformar una base de datos desordenada a formato `tidy`, en R se puede hacer uso de los paquetes `dplyr` y `tidyr`.

Para nuestro caso de interés, los paquetes `tidyxl` y `unpivotr`, que fueron usados para el procesamiento avanzado de bases de datos en *excel*, se basan fuertemente en importar las bases de datos en excel a `R` en formato `tidy` tal cual se ha mencionado en esta sesión.

4.1.2 Expresiones regulares (Regex)

4.2 Base de datos necesaria para realizar el procesamiento automático de los datos del 3PRF

Dentro del directorio de trabajo `input_web_scraping` se encuentra la base de datos `input_web_scraping` cuya pestaña `procesamiento_bases` contiene la base de datos necesaria para realizar el procesamiento automático de las bases del 3PRF para las variables que han sido automatizadas, que por el momento son 46 de las 72 variables del modelo. El contenido de dicha base de datos se encuentra en la Tabla 4.

Tabla 4: Tabla de las variables del modelo 3PRF cuyo procesamiento ya fue automatizado

nombre_variable	nombre_base	nombre_sheet	palabra_identificacion_tabla	Fuente
ise	DANE_ISE_9_actividades.xlsx	Cuadro 2	Concepto	DANE
prod_industrial	DANE_EMMET_territorial.xlsx	8. Índices Desestacionalizados	Dominios	DANE
ventas_reales	DANE_EMC.xlsx	3.1	Año	DANE
impo_total_usd	DANE_Importaciones.xlsx	Cuadro A13	CUODE	DANE
impo_consumo	DANE_Importaciones.xlsx	Cuadro A13	CUODE	DANE
impo_intermedios	DANE_Importaciones.xlsx	Cuadro A13	CUODE	DANE
impo_bienescapital	DANE_Importaciones.xlsx	Cuadro A13	CUODE	DANE
expo_total_usd	DANE_Exportaciones.xlsx	Tra y Notra	MES	DANE
expo_tradicionales	DANE_Exportaciones.xlsx	Tra y Notra	MES	DANE
expo_no_tradicionales	DANE_Exportaciones.xlsx	Tra y Notra	MES	DANE
despachos_cemento	DANE_ECG.xlsx	Anexo 1	Año	DANE
area_construccion	DANE_ELIC.xlsx	1. Área aprobada	Años y meses	DANE
ipp	DANE_IPP.xlsx	IPP Histórico	Producción Nacional12	DANE
ipc	DANE_IPC.xlsx	IndicesIPC	Mes	DANE
ingresos_reales_hoteles	DANE_EMA.xlsx	Desestacionalización	Año	DANE
td_nacional	DANE_Mercado_laboral.xlsx	Tnal mensual	Concepto	DANE
to_nacional	DANE_Mercado_laboral.xlsx	Tnal mensual	Concepto	DANE
ocupados	DANE_Mercado_laboral.xlsx	Tnal mensual	Concepto	DANE
desocupados	DANE_Mercado_laboral.xlsx	Tnal mensual	Concepto	DANE
ganado_vacuno_volu	DANE_Sacrificio_ganado.xlsx	Cuadro_1	Periodo	DANE
ganado_porcino_volum	DANE_Sacrificio_ganado.xlsx	Cuadro_3	Periodo	DANE
produccion_carbon_volum	DANE_IPI.xlsx	3. Indices total por clase	Dominios	DANE
expo_carbon_volum	DANE_Exportaciones.xlsx	Tra y Notra	MES	DANE
empleo_industrial	DANE_EMMET_territorial.xlsx	8. Índices Desestacionalizados	Dominios	DANE
empleo_minorista	DANE_EMC.xlsx	3.1	Año	DANE
lanzamientos_viv	CAMACOL_Tablas_coyuntura.xlsx	Lanzamientos	Fecha	CAMACOL
iniciaciones_viv	CAMACOL_Tablas_coyuntura.xlsx	Iniciaciones	Fecha	CAMACOL
ventas_viv	CAMACOL_Tablas_coyuntura.xlsx	Ventas	Fecha	CAMACOL
transporte_pasajeros_aereos	AEROCIVIL_Estadisticas.xlsx		Fecha	AEROCIVIL
transporte_aereo_carga	AEROCIVIL_Estadisticas.xlsx		Fecha	AEROCIVIL
produccion_cafe	FNC_Estadisticas.xlsx	9. Producción mensual	Mes	FNC
precio_cafe	FNC_Estadisticas.xlsx	3. Precio Ex_Dock Mensual	Mes	FNC
iva_interno			Año	DIAN
impuestos_externos			Año	DIAN
ipc_core	Banrep_Inflacion_basica.xlsx		Núcleo 15	BANREP
ipc_alimentos	Banrep_Inflacion_basica.xlsx		Núcleo 15	BANREP
ipc_regulados	Banrep_Medidas_inflacion.xlsx		Alimentos	BANREP
tasa_cambio_real	Banrep_tasa_cambio_real.xlsx		ITCR no tradicional (NT) (1)	BANREP
tasa_cambio_nominal	Banrep_tasa_cambio_nominal.xlsx		Año - Mes (aaaa -mm)	BANREP
cero_cupon_1	Banrep_tasa_cero_cupon_pesos.xlsx		Año(aaaa)-Mes(mm)	BANREP
cero_cupon_5	Banrep_tasa_cero_cupon_pesos.xlsx		Año(aaaa)-Mes(mm)	BANREP
cero_cupon_10	Banrep_tasa_cero_cupon_pesos.xlsx		Año(aaaa)-Mes(mm)	BANREP
cartera_consumo	Banrep_Cartera_sistema_financiero.xlsx		Cartera Comercial Tradicional 1	BANREP
cartera_comercial	Banrep_Cartera_sistema_financiero.xlsx		Cartera Comercial Tradicional 1	BANREP
cartera_hipotecaria	Banrep_Cartera_bruta_y_neta.xlsx		Cartera Bruta por Tipo	BANREP
remesas_usd	Banrep_Remesas_historico.xlsx		Año(aaaa)-Mes(mm)	BANREP

4.3 Paquetes para realizar el procesaiento automático de las bases de datos del 3PRF en R

4.3.1 Procesamiento avanzado de datos de Excel en R: `tidyxl` y `unpivotr`

4.3.2 Expresiones regulares en R: `stringr`

Funciones principales del paquete stringr para el manejo de expresiones regulares en R

- str_detect:
  - `str_detect(string, pattern, negate = FALSE)`:
    - Input:
      - `string`: Vector de carácteres
      - `pattern`: Expresión regular
    - output:
      - Retorna un vector `lógico` que indica si `pattern` (*la expresión regular*) se encuentra o no en cada uno de los elementos del vector `string`
- str_extract:
  - `str_extract(string, pattern, group = NULL)`
    - Input:
      - `string`: Vector de carácteres
      - `pattern`: Expresión regular
    - Output:
      - Extrae para cada elemento del vector `string` el primer "match" completo que contenga la expresión regular `pattern`
- str_sub:
  - `str_sub(string, start = 1L, end = -1L)`:
    - Input:
      - `string`: Vector de carácteres
    - Output:
      - Extrae los elementos del vector `string` que se encuentren en las posiciones específicadas por `start` y `end`

4.4 Código para realizar el procesamiento automático de los datos del 3PRF

4.4.1 Función para la extracción del primer mes de cada base

```
# Función para la extracción del primer mes de cada base
deteccion_primer_mes = function(mes_text){

  if(str_detect(mes_text, regex("enero|ene", ignore_case = TRUE))){

    mes = 1

  }else if(str_detect(mes_text, regex("febrero|feb", ignore_case = TRUE))){

    mes = 2

  }else if(str_detect(mes_text, regex("marzo|mar", ignore_case = TRUE))){

    mes = 3

  }else if(str_detect(mes_text, regex("abril|abr", ignore_case = TRUE))){

    mes = 4

  }else if(str_detect(mes_text, regex("mayo|may", ignore_case = TRUE))){
```

```
    mes = 5

  }else if(str_detect(mes_text, regex("junio|jun", ignore_case = TRUE))){

    mes = 6

  }else if(str_detect(mes_text, regex("julio|jul", ignore_case = TRUE))){

    mes = 7

  }else if(str_detect(mes_text, regex("agosto|ago", ignore_case = TRUE))){

    mes = 8

  }else if(str_detect(mes_text, regex("septiembre|sep", ignore_case = TRUE))){

    mes = 9

  }else if(str_detect(mes_text, regex("octubre|oct", ignore_case = TRUE))){

    mes = 10

  }else if(str_detect(mes_text, regex("noviembre|nov", ignore_case = TRUE))){

    mes = 11

  }else if(str_detect(mes_text, regex("diciembre|dic", ignore_case = TRUE))){

    mes = 12

  }

  return(mes)

}
```

4.5 Funciones para el procesamiento automático en R de las diferentes bases

4.5.0.1 Función para el procesamiento de las bases de datos del DANE

```
# Función para procesar las bases del DANE
procesamiento_DANE = function(nombre_base, nombre_sheet, palabra_identificacion_tabla, nombre_variable){

  # Imporatación de la base de datos mediante la función "xlsx_cells"
  base = tidyxl::xlsx_cells(nombre_base, sheets = nombre_sheet)


  # Identificación de la esquina superior izquierda de la tabla
  esquina_sup_izq_tabla = dplyr::filter(base, character == palabra_identificacion_tabla)

  # Particion de la base que contiene la informacion de la tabla de interés dentro de la hoja de excel
  particion = partition(base, esquina_sup_izq_tabla)

  # ==== Inicio: Específico para cada base ====

  # La selección de la información cambia dependiendo de la base de datos
  if (nombre_base == "DANE_ISE_9_actividades.xlsx"){

    # 2.1 DANE_ISE_9_actividades.xlsx ----

    # Código para la base de datos que contiene a la variable "ise"

    # Selección de las celdas de Excel que contienen la info de la tabla
    tabla = particion$cells[[1]] %>%
      filter(!is_blank) %>%
```

```
    behead("up-left", "año") %>%
    behead("up", "mes") %>%
    behead("left", "actividad") %>%
    select(row, col, data_type, numeric, character, date, año, mes, actividad)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(actividad == "Indicador de Seguimiento a la Economía") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = tabla_selection$año[1]

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla_selection$mes[1])

}else if(nombre_base == "DANE_EMMET_territorial.xlsx"){

  # 2.2 DANE_EMMET_territorial.xlsx ----

  # Código para las variables de la base "EMMET" (Industria)

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados") %>%
    select(row, col, data_type, numeric, character, date, encabezados)

  # La selección de la información cambia dependiendo de la "variable de industría"
  if (nombre_variable == "prod_industrial"){

    # Código para la selección de la variable de "prod_industrial"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "Producción \r\nreal ") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "empleo_industrial"){

    # Código para la selección de la variable de "empleo_industrial"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "Empleo  Total") %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio

  ## Año inicial
  tabla_year_init = tabla %>% filter(encabezados == "Año")
  year_init = tabla_year_init$numeric[1]

  ## Mes inicial
  tabla_mes_init = tabla %>% filter(encabezados == "Mes")
  mes_init = tabla_mes_init$numeric[1]

}else if(nombre_base == "DANE_EMC.xlsx"){

  # 2.3 DANE_EMC.xlsx ----

  # Código para las variables de la base "EMC" (Comercio)
```

```
  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados") %>%
    select(row, col, data_type, numeric, character, date, encabezados)

  # La selección de la información cambia dependiendo de la "variable de comercio"
  if (nombre_variable == "ventas_reales"){

    # Código para la selección de la variable de "ventas_reales"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "Total Comercio Minorista sin Combustibles ni Vehículosd") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "empleo_minorista"){

    # Código para la selección de la variable de "empleo_minorista"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "Total Personal Ocupado") %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio

  ## Año inicial
  tabla_year_init = tabla %>% filter(encabezados == "Año")
  year_init = tabla_year_init$numeric[1]

  ## Mes inicial
  tabla_mes_init = tabla %>% filter(encabezados == "Mes")
  mes_init = deteccion_primer_mes(tabla_mes_init$character[1])

}else if(nombre_base == "DANE_Importaciones.xlsx"){

  # 2.4 DANE_Importaciones.xlsx ----

  # Se filtra la base para que solo se seleccione la celda que tiene la información "Mes (Año-Año)p"
  base_init_year_mes = base %>%
    filter(str_detect(base$character, regex(".*\\(\\d{4} - \\d{4}\\).*", ignore_case = TRUE)))

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = str_extract(base_init_year_mes$character, regex("\\d{4}"))

  ## Mes inicial
  mes_init = deteccion_primer_mes(str_extract(base_init_year_mes$character, regex("[A-Za-z]+")))

  # Código para la base de datos que contiene las variables de "importación"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up-left", "encabezados2") %>%
    behead("left", "CUODE") %>%
    behead("left", "descripcion_importacion") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2, CUODE, descripcion_importacion)
```

```r
# La selección de la información cambia dependiendo de la "variable de importación"
if (nombre_variable == "impo_total_usd"){

  # Código para la selección de la variable de "impo_total_usd"

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(descripcion_importacion == "Total importaciones") %>%
    filter(encabezados1 == paste0(year_init, "p")) %>%
    filter(!is.na(numeric))

}else if (nombre_variable == "impo_consumo"){

  # Código para la selección de la variable de "impo_consumo"

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(descripcion_importacion == "Bienes de consumo") %>%
    filter(encabezados1 == paste0(year_init, "p")) %>%
    filter(!is.na(numeric))

}else if (nombre_variable == "impo_intermedios"){

  # Código para la selección de la variable de "impo_intermedios"

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(descripcion_importacion == "Materias primas y productos intermedios") %>%
    filter(encabezados1 == paste0(year_init, "p")) %>%
    filter(!is.na(numeric))

}else if (nombre_variable == "impo_bienescapital"){

  # Código para la selección de la variable de "impo_bienescapital"

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(descripcion_importacion == "Bienes de capital y material de construcción") %>%
    filter(encabezados1 == paste0(year_init, "p")) %>%
    filter(!is.na(numeric))

}

# Dependiendo de si la base es de Enero u otra mes del año, el archivo Excel cambia
if (nrow(tabla_selection) == 1){

  # Creación del objeto ts
  ts_obj = ts(tabla_selection[1,]$numeric, start = c(year_init, mes_init), frequency = 12)

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
  return(xts_obj)

}else if (nrow(tabla_selection) == 2){

  # Creación del objeto ts
  ts_obj = ts(tabla_selection[2,]$numeric, start = c(year_init, mes_init), frequency = 12)

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
```

```
    return(xts_obj)

  }

}else if(nombre_base == "DANE_Exportaciones.xlsx"){

  # 2.5 DANE_Exportaciones.xlsx ----

  # Código para la base de datos que contiene las variables de "exportación"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "tipo_exportacion") %>%
    behead("up-left", "producto_exportacion") %>%
    behead("up", "metrica_exportacion") %>%
    behead("left", "mes") %>%
    select(row, col, data_type, numeric, character, date, tipo_exportacion, producto_exportacion, metrica_exportacion, mes)

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = str_extract(tabla$mes[1], regex("\\d{4}"))

  ## Mes inicial
  mes_init = as.numeric(str_extract_all(tabla$mes[1], regex("\\d{2}"))[[1]][3]) # Toca hacer todo esto para lograr extraer

  # La selección de la información cambia dependiendo de la "variable de exportación"
  if (nombre_variable == "expo_total_usd"){

    # Código para la selección de la variable de "expo_total_usd"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(tipo_exportacion == "Total exportaciones") %>%
      filter(metrica_exportacion == "Miles de Dólares FOB") %>%
      filter(!str_detect(mes, "Totales")) %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "expo_tradicionales"){

    # Código para la selección de la variable de "expo_tradicionales"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(tipo_exportacion == "Exportaciones tradicionales") %>%
      filter(producto_exportacion == "Total Exportaciones Tradicionales ") %>%
      filter(metrica_exportacion == "Miles de Dólares FOB") %>%
      filter(!str_detect(mes, "Totales")) %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "expo_no_tradicionales"){

    # Código para la selección de la variable de "expo_no_tradicionales"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(tipo_exportacion == "Exportaciones no tradicionales") %>%
      filter(metrica_exportacion == "Miles de Dólares FOB") %>%
      filter(!str_detect(mes, "Totales")) %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "expo_carbon_volum"){

    # Código para la selección de la variable de "expo_no_tradicionales"
```

```
    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(tipo_exportacion == "Exportaciones tradicionales") %>%
      filter(producto_exportacion == "Carbón") %>%
      filter(metrica_exportacion == "Toneladas Métricas") %>%
      filter(!str_detect(mes, "Totales")) %>%
      filter(!is.na(numeric))

  }

}else if(nombre_base == "DANE_ECG.xlsx"){

  # 2.6 DANE_ECG.xlsx ----

  # Código para la base de datos que contiene a la variable "despachos_cemento"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up", "encabezados2") %>%
    behead("left-up", "año") %>%
    behead("left", "mes") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2, año, mes)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezados1 == "Toneladas ") %>%
    filter(encabezados2 == "Despachos Nacionales") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = tabla_selection$año[1]

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla_selection$mes[1])

}else if(nombre_base == "DANE_ELIC.xlsx"){

  # 2.7 DANE_ELIC.xlsx ----

  # Código para la base de datos que contiene a la variable "area_construccion"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up", "encabezados2") %>%
    behead("left-up", "año") %>%
    behead("left", "mes") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2, año, mes)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezados1 == "Total") %>%
    filter(encabezados2 == "302 \r\nmunicipios") %>%
    filter(!is.na(numeric))

  # Promedio area construida año 2015
  promedio_area_construida_2015 = 2616877

  # Estandarización de la variable
  tabla_final = tabla_selection %>%
    mutate(area_construccion =  (numeric * 100) / promedio_area_construida_2015)
```

```
  # Selección del año y mes de inicio

  ## Año inicial
  year_init = tabla_final$año[1]

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla_final$mes[1])

  # Creación del objeto ts
  ts_obj = ts(tabla_final$area_construccion, start = c(year_init, mes_init), frequency = 12)

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
  return(xts_obj)

}else if(nombre_base == "DANE_IPP.xlsx"){

  # 2.8 DANE_IPP.xlsx ----

  # Código para la base de datos que contiene a la variable "despachos_cemento"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezado1") %>%
    behead("up", "encabezado2") %>%
    select(row, col, data_type, numeric, character, date, encabezado1, encabezado2)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezado1 == "Producción Nacional12") %>%
    filter(encabezado2 == "Producción Nacional") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = na.omit(str_extract(base$character, regex("\\d{4}")))[2]

  # Esquina superior izquierda de la tabla
  esquina_sup_izq_tabla2 = dplyr::filter(base, character == year_init)

  # Particion de la base que contiene la informacion de la tabla de interés dentro de la hoja de excel
  particion2 = partition(base, esquina_sup_izq_tabla2)

  tabla2 = particion2$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("left-up", "año") %>%
    behead("left", "mes") %>%
    select(row, col, data_type, numeric, character, date, año, mes)

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla2$mes[1])

}else if(nombre_base == "DANE_IPC.xlsx"){

  # 2.9 DANE_IPC.xlsx ----

  # Código para la base de datos que contiene a la variable "despachos_cemento"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
```

```
    filter(!is_blank) %>%
    behead("up", "año") %>%
    behead("left", "mes") %>%
    select(row, col, data_type, numeric, character, date, año, mes)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    arrange(año) %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = tabla_selection$año[1]

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla_selection$mes[1])

}else if(nombre_base == "DANE_EMA.xlsx"){

  # 2.10 DANE_EMA.xlsx ----

  # Código para la base de datos que contiene a la variable "despachos_cemento"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados") %>%
    behead("left-up", "año") %>%
    behead("left", "mes") %>%
    select(row, col, data_type, numeric, character, date, encabezados, año, mes)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezados == "Ingresos totales reales") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = tabla_selection$año[1]

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla_selection$mes[1])

}else if(nombre_base == "DANE_Mercado_laboral.xlsx"){

  # 2.11 DANE_Mercado_laboral.xlsx ----

  # Código para la base de datos que contiene las variables de "mercado laboral"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "año") %>%
    behead("up", "mes") %>%
    behead("left", "concepto_mercado_laboral") %>%
    select(row, col, data_type, numeric, character, date, año, mes, concepto_mercado_laboral)

  # La selección de la información cambia dependiendo de la "variable de mercado laboral"
  if (nombre_variable == "td_nacional"){

    # Código para la selección de la variable de "td_nacional"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
```

```
      filter(concepto_mercado_laboral == "Tasa de Desocupación (TD)") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "to_nacional"){

    # Código para la selección de la variable de "to_nacional"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(concepto_mercado_laboral == "Tasa de Ocupación (TO)") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "ocupados"){

    # Código para la selección de la variable de "ocupados"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(concepto_mercado_laboral == "Población ocupada") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "desocupados"){

    # Código para la selección de la variable de "desocupados"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(concepto_mercado_laboral == "Población desocupada") %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = tabla$año[1]

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla$mes[1])

}else if(nombre_base == "DANE_Sacrificio_ganado.xlsx"){

  # 2.12 DANE_Sacrificio_ganado.xlsx ----

  # Código para la base de datos que contiene las variables de "Sacrificio de ganado"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up", "encabezados2") %>%
    behead("left", "periodo") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2, periodo)

  # La selección de la información cambia dependiendo de la "variable de mercado laboral"
  if (nombre_variable == "ganado_vacuno_volu"){

    # Código para la selección de la variable de "ganado_vacuno_volu"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Total general1") %>%
      filter(encabezados2 == "Peso en pie     (kilos)") %>%
      filter(periodo != "Total general") %>%
      filter(!is.na(numeric))
```

```
  }else if(nombre_variable == "ganado_porcino_volum"){

    # Código para la selección de la variable de "ganado_porcino_volum"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Total general") %>%
      filter(encabezados2 == "Peso en pie     (kilos)") %>%
      filter(periodo != "Total general") %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = na.omit(str_extract(base$character, regex("\\d{4}")))[1]

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla_selection$periodo[1])

}else if(nombre_base == "DANE_IPI.xlsx"){

  # 2.13 DANE_IPI.xlsx ----

  # Código para la base de datos que contiene a la variable "produccion_carbon_volum"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados") %>%
    behead("left", "dominios") %>%
    behead("left", "año") %>%
    behead("left", "mes") %>%
    behead("left", "clases_industriales") %>%
    select(row, col, data_type, numeric, character, date, encabezados, año, mes, clases_industriales)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(clases_industriales == "Extracción de hulla (carbón de piedra)") %>%
    filter(!is.na(numeric))


  # Selección del año y mes de inicio

  ## Año inicial
  year_init = tabla_selection$año[1]

  ## Mes inicial
  mes_init = tabla_selection$mes[1]

}

# ==== Fin: Específico para cada base ====

if (!(nombre_base %in% c("DANE_Importaciones.xlsx", "DANE_ELIC.xlsx"))){

  # Creación del objeto ts
  ts_obj = ts(tabla_selection$numeric, start = c(year_init, mes_init), frequency = 12)

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
  return(xts_obj)
```

```
  }

}
```

4.5.0.2 Función para el procesamiento de las bases de datos de Camacol

```
# Función para procesar las bases de Camacol
procesamiento_Camacol = function(nombre_base, nombre_sheet, palabra_identificacion_tabla, nombre_variable){

  # Importacion base de datos
  base = tidyxl::xlsx_cells(nombre_base, sheets = nombre_sheet) %>%
    select(row, col, data_type, numeric, character, date)

  # Esquina superior izquierda de la tabla
  esquina_sup_izq_tabla = dplyr::filter(base, character == palabra_identificacion_tabla)

  # Particion de la base que contiene la informacion de la tabla de interés dentro de la hoja de excel
  particion = partition(base, esquina_sup_izq_tabla)

  # Tabla de datos con la información de interés
  tabla = particion$cells[[1]] %>%
    behead("NNW", "regiones") %>%
    behead("N", "tipo_de_vivienda")

  # Tabla de datos filtrada solo con la variable de interés
  tabla_selection = tabla %>%
    filter(regiones == "13 Regionales") %>%
    filter(tipo_de_vivienda == "TOTAL") %>%
    filter(!is.na(numeric))

  # Para que no me seleccione las últimas 6 filas, que no hacen parte de la tabla
  tabla_selection = head(tabla_selection, nrow(tabla_selection) - 6)

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = year(na.omit(tabla$date)[1])

  ## Mes inicial
  mes_init = month(na.omit(tabla$date)[1])

  # Creación del objeto ts
  ts_obj = ts(tabla_selection$numeric, start = c(year_init, mes_init), frequency = 12)

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
  return(xts_obj)

}
```

4.5.0.3 Función para el procesamiento de las bases de datos de la Aerocivil

```
# Función para procesar las bases de la Aerocivil
procesamiento_Aerocivil = function(nombre_base, nombre_sheet, palabra_identificacion_tabla, nombre_variable){

  # Importacion base de datos
  base = tidyxl::xlsx_cells(nombre_base)

  # Esquina superior izquierda de la tabla
  esquina_sup_izq_tabla = dplyr::filter(base, character == palabra_identificacion_tabla)
```

```
  # Particion de la base que contiene la informacion de la tabla de interés dentro de la hoja de excel
  particion = partition(base, esquina_sup_izq_tabla)

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados") %>%
    select(row, col, data_type, numeric, character, date, encabezados)

  # Selección del año y mes de inicio

  ## Año inicial
  year_init =  year(na.omit(tabla$date)[1])

  ## Mes inicial
  mes_init = month(na.omit(tabla$date)[1])

  # La selección de la información cambia dependiendo de la variable de la "aeorocivil"
  if (nombre_variable == "transporte_pasajeros_aereos"){

    # Código para la selección de la variable de "transporte_pasajeros_aereos"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "Pasajeros") %>%
      filter(!is.na(numeric))

    # Suma de los valores de la columna de interés
    tabla_sum = tabla_selection %>%
      summarize(sum_pasajeros = sum(numeric))

    # Creación del objeto ts
    ts_obj = ts(tabla_sum$sum_pasajeros, start = c(year_init, mes_init), frequency = 12)

  }else if(nombre_variable == "transporte_aereo_carga"){

    # Código para la selección de la variable de "transporte_aereo_carga"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "Carga + Correo (Kg)") %>%
      filter(!is.na(numeric))

    # Suma de los valores de la columna de interés
    tabla_sum = tabla_selection %>%
      summarize(sum_carga_correo = sum(numeric))

    # Creación del objeto ts
    ts_obj = ts(tabla_sum$sum_carga_correo, start = c(year_init, mes_init), frequency = 12)

  }

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
  return(xts_obj)

}
```

4.5.0.4 Función para el procesamiento de las bases de datos de la FNC

```
# Función para procesar las bases de la FNC
procesamiento_FNC = function(nombre_base, nombre_sheet, palabra_identificacion_tabla, nombre_variable){
```

```
  # Importacion base de datos
  base = tidyxl::xlsx_cells(nombre_base, sheets = nombre_sheet)

  # Esquina superior izquierda de la tabla
  esquina_sup_izq_tabla = dplyr::filter(base, character == palabra_identificacion_tabla)

  # Particion de la base que contiene la informacion de la tabla de interés dentro de la hoja de excel
  particion = partition(base, esquina_sup_izq_tabla)

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados") %>%
    behead("left", "fecha") %>%
    select(row, col, data_type, numeric, character, date, encabezados, fecha)


  # La selección de la información cambia dependiendo de la variable de la "FNC"
  if (nombre_variable == "produccion_cafe"){

    # Código para la selección de la variable de "produccion_cafe"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "Producción") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "precio_cafe"){

    # Código para la selección de la variable de "precio_cafe"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "Precio externo") %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = year(na.omit(tabla_selection$fecha)[1])

  ## Mes inicial
  mes_init = month(na.omit(tabla_selection$fecha)[1])

  # Creación del objeto ts
  ts_obj = ts(tabla_selection$numeric, start = c(year_init, mes_init), frequency = 12)

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
  return(xts_obj)

}
```

4.5.0.5 Función para el procesamiento de las bases de datos de la DIAN

```
# Función para procesar las bases de la DIAN
procesamiento_DIAN = function(nombre_base, nombre_sheet, palabra_identificacion_tabla, nombre_variable){

  # Importacion base de datos
```

```
  base = tidyxl::xlsx_cells(nombre_base)

  # Esquina superior izquierda de la tabla
  esquina_sup_izq_tabla = dplyr::filter(base, character == palabra_identificacion_tabla)

  # Particion de la base que contiene la informacion de la tabla de interés dentro de la hoja de excel
  particion = partition(base, esquina_sup_izq_tabla)

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados") %>%
    behead("left", "año") %>%
    behead("left", "mes") %>%
    select(row, col, data_type, numeric, character, date, encabezados, año, mes)


  # La selección de la información cambia dependiendo de la variable de la "FNC"
  if (nombre_variable == "iva_interno"){

    # Código para la selección de la variable de "iva_interno"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "2. IVA interno (2.1+2.2)") %>%
      filter(!str_detect(año, "TOTAL")) %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "impuestos_externos"){

    # Código para la selección de la variable de "impuestos_externos"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados == "B. Externos (15+16)") %>%
      filter(!str_detect(año, "TOTAL")) %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio

  ## Año inicial
  year_init = tabla_selection$año[1]

  ## Mes inicial
  mes_init = deteccion_primer_mes(tabla_selection$mes[1])

  # Creación del objeto ts
  ts_obj = ts(tabla_selection$numeric, start = c(year_init, mes_init), frequency = 12)

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
  return(xts_obj)

}
```

4.5.0.6 Función para el procesamiento de las bases de datos del Banrep

```
# 7. Función para el procesamiento de las bases descargadas del Banrep ------------------------------------------------
procesamiento_Banrep = function(nombre_base, nombre_sheet, palabra_identificacion_tabla, nombre_variable){
```

```r
# Variable "flag" que contrala si hay que invertir el orden de un vector o no
rev_vect = 0 # 0 no hay que revertir, 1 sí hay que revertir

# Imporatación de la base de datos mediante la función "xlsx_cells"
base = tidyxl::xlsx_cells(nombre_base)

# Identificación de la esquina superior izquierda de la tabla
esquina_sup_izq_tabla = dplyr::filter(base, character == palabra_identificacion_tabla)

# Particion de la base que contiene la informacion de la tabla de interés dentro de la hoja de excel
particion = partition(base, esquina_sup_izq_tabla)

# ==== Inicio: Específico para cada base ====

# La selección de la información cambia dependiendo de la base de datos
if(nombre_base == "Banrep_Inflacion_basica.xlsx"){

  # 7.1 Banrep_Inflacion_basica.xlsx ----

  # Código para la base de datos que contiene las variables de "inflación básica y de alimentos"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[2]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up", "encabezados2") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2)

  # La selección de la información cambia dependiendo de la "variable de inflación"
  if (nombre_variable == "ipc_core"){

    # Código para la selección de la variable de "ipc_core"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Núcleo 15") %>%
      filter(encabezados2 == "Índice") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "ipc_alimentos"){

    # Código para la selección de la variable de "ipc_alimentos"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Alimentos") %>%
      filter(encabezados2 == "Índice") %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio

  # Se filtra la base de datos para que solo seleccione las observaciones que tengan fecha
  fechas = base %>%
    filter(!is.na(date))

  ## Año inicial
  year_init = str_extract(fechas$date[1], "\\d{4}")

  ## Mes inicial
  mes_init = str_extract(fechas$date[1], "(?<=-)\\d{2}")

}else if (nombre_base == "Banrep_Medidas_inflacion.xlsx"){

  # 7.2 Banrep_Medidas_inflacion.xlsx ----
```

```
  # Código para la base de datos que contiene a la variable "ipc_regulados"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[2]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up", "encabezados2") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezados1 == "Regulados") %>%
    filter(encabezados2 == "Índice") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  # Se filtra la base de datos para que solo seleccione las observaciones que tengan fecha
  fechas = base %>%
    filter(!is.na(date))

  ## Año inicial
  year_init = str_extract(fechas$date[1], "\\d{4}")

  ## Mes inicial
  mes_init = str_extract(fechas$date[1], "(?<=-)\\d{2}")

}else if (nombre_base == "Banrep_tasa_cambio_real.xlsx"){

  # 7.3 Banrep_tasa_cambio_real.xlsx ----

  # Código para la base de datos que contiene a la variable "tasa_cambio_real"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up", "encabezados2") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezados1 == "ITCR comercio total (T) (2)") %>%
    filter(encabezados2 == "ITCR_IPC(T) (6)") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  ## Año inicial
  year_init =  str_sub(na.omit(str_extract(base$numeric, "\\b\\d{6}\\b"))[1], 1, 4)

  ## Mes inicial
  mes_init = str_sub(na.omit(str_extract(base$numeric, "\\b\\d{6}\\b"))[1], 5, 6)

}else if (nombre_base == "Banrep_tasa_cambio_nominal.xlsx"){

  # 7.4 Banrep_tasa_cambio_nominal.xlsx ----

  # Hay que revertir la variable
  rev_vect = 1

  # Código para la base de datos que contiene a la variable "tasa_cambio_nominal"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
```

```
    filter(!is_blank) %>%
    behead("up", "encabezados1") %>%
    select(row, col, data_type, numeric, character, date, encabezados1)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezados1 == "Promedio mensual") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  # Indica el índice de la última observación del vector de interés
  num_fechas = length(na.omit(tabla$date))

  ## Año inicial
  year_init = str_extract(na.omit(tabla$date)[num_fechas], "\\d{4}")

  ## Mes inicial
  mes_init = str_extract(na.omit(tabla$date)[num_fechas], "(?<=-)\\d{2}")

}else if(nombre_base == "Banrep_tasa_cero_cupon_pesos.xlsx"){

  # 7.5 Banrep_tasa_cero_cupon_pesos.xlsx ----

  # Hay que revertir la variable
  rev_vect = 1

  # Código para la base de datos que contiene las variables de las "tasas cero cupón"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados1") %>%
    select(row, col, data_type, numeric, character, date, encabezados1)

  # La selección de la información cambia dependiendo de la "variable de tasa cero cupón"
  if (nombre_variable == "cero_cupon_1"){

    # Código para la selección de la variable de "cero_cupon_1"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Tasa a 1 año %") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "cero_cupon_5"){

    # Código para la selección de la variable de "cero_cupon_5"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Tasa a 5 años %") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "cero_cupon_10"){

    # Código para la selección de la variable de "cero_cupon_10"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Tasa a 10 años %") %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio
```

```r
  # Indica el índice de la última observación del vector de interés
  num_fechas = length(na.omit(tabla$date))

  ## Año inicial
  year_init = str_extract(na.omit(tabla$date)[num_fechas], "\\d{4}")

  ## Mes inicial
  mes_init = str_extract(na.omit(tabla$date)[num_fechas], "(?<=-)\\d{2}")

}else if(nombre_base == "Banrep_Cartera_sistema_financiero.xlsx"){

  # 7.6 Banrep_Cartera_sistema_financiero.xlsx ----

  # Hay que revertir la variable
  rev_vect = 1

  # Código para la base de datos que contiene las variables de la "cartera del sistemas financiero"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up", "encabezados2") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2)

  # La selección de la información cambia dependiendo de la "variable de cartera del sistemas financiero"
  if (nombre_variable == "cartera_consumo"){

    # Código para la selección de la variable de "cartera_consumo"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Cartera Consumo Tradicional") %>%
      filter(!is.na(numeric))

  }else if(nombre_variable == "cartera_comercial"){

    # Código para la selección de la variable de "cartera_comercial"

    # Selección de la columna con la información de interés
    tabla_selection = tabla %>%
      filter(encabezados1 == "Cartera Comercial Tradicional 1") %>%
      filter(!is.na(numeric))

  }

  # Selección del año y mes de inicio

  # Se filtra la base de datos para que solo seleccione las observaciones que tengan fecha
  fechas = base %>%
    filter(!is.na(date))

  # Indica el índice de la última observación del vector de interés
  num_fechas = length(na.omit(fechas$date))

  ## Año inicial
  year_init = str_extract(fechas$date[num_fechas], "\\d{4}")

  ## Mes inicial
  mes_init = str_extract(fechas$date[num_fechas], "(?<=-)\\d{2}")

}else if (nombre_base == "Banrep_Cartera_bruta_y_neta.xlsx"){

  # 7.7 Banrep_Cartera_bruta_y_neta.xlsx ----
```

```
  # Hay que revertir la variable
  rev_vect = 1

  # Código para la base de datos que contiene a la variable "cartera_hipotecaria"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up-left", "encabezados1") %>%
    behead("up-left", "encabezados2") %>%
    behead("up", "encabezados3") %>%
    select(row, col, data_type, numeric, character, date, encabezados1, encabezados2, encabezados3)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezados1 == "Cartera Bruta por Tipo") %>%
    filter(encabezados2 == "Cartera Hipotecaria ajustada") %>%
    filter(encabezados3 == "Legal") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  # Se filtra la base de datos para que solo seleccione las observaciones que tengan fecha
  fechas = base %>%
    filter(!is.na(date))

  # Indica el índice de la última observación del vector de interés
  num_fechas = length(na.omit(fechas$date))

  ## Año inicial
  year_init = str_extract(fechas$date[num_fechas], "\\d{4}")

  ## Mes inicial
  mes_init = str_extract(fechas$date[num_fechas], "(?<=-)\\d{2}")

}else if (nombre_base == "Banrep_Remesas_historico.xlsx"){

  # 7.8 Banrep_Remesas_historico.xlsx ----

  # Hay que revertir la variable
  rev_vect = 1

  # Código para la base de datos que contiene a la variable "remesas_usd"

  # Selección de las celdas de Excel que contienen la info de la tabla
  tabla = particion$cells[[1]] %>%
    filter(!is_blank) %>%
    behead("up", "encabezados1") %>%
    select(row, col, data_type, numeric, character, date, encabezados1)

  # Selección de la columna con la información de interés
  tabla_selection = tabla %>%
    filter(encabezados1 == "Valor en millones de dólares estadounidenses") %>%
    filter(!is.na(numeric))

  # Selección del año y mes de inicio

  # Indica el índice de la última observación del vector de interés
  num_fechas = length(na.omit(tabla$date))

  ## Año inicial
  year_init = str_extract(na.omit(tabla$date)[num_fechas], "\\d{4}")

  ## Mes inicial
  mes_init = str_extract(na.omit(tabla$date)[num_fechas], "(?<=-)\\d{2}")
```

```
  }

  # ==== Fin: Específico para cada base ====

  # Creación del objeto ts

  if (rev_vect == 0){

    # Condición si no hay que invertir el vector
    ts_obj = ts(tabla_selection$numeric, start = c(year_init, mes_init), frequency = 12)

  }else if(rev_vect == 1){

    # Condición en caso de que haya que invertir el vector
    ts_obj = ts(rev(tabla_selection$numeric), start = c(year_init, mes_init), frequency = 12)

  }

  # Creación del objeto xts
  xts_obj = as.xts(ts_obj)
  colnames(xts_obj) = c(nombre_variable)

  # Retorna el objeto xts
  return(xts_obj)

}
```

4.5.1 Funciones para el procesamiento de todas las bases de datos vía la base de datos `input_web_scraping`

4.5.1.1 Función auxiliar "tryCatch"

```
funcion_try_catch = function(funcion_procesamiento_base, nombre_base, nombre_sheet, ...){

  # Nota:
  ### Try: Nombre de la "Sheet" de excel no tiene espacio al final del nombre de la hoja de excel
  ### Catch: Nombre de la "Sheet" de excel tiene espacio al final del nombre de la hoja de excel

  tryCatch({

    # Función que procesa las bases de datos cuándo
    xts_obj = funcion_procesamiento_base(nombre_base, nombre_sheet, ...)

    return(xts_obj)

  }, error = function(e) {
    # Código que se ejecuta cuando no se puede desestacionalizar por default
    # En este caso se aplica la opción "outlier = NULL"

    nombre_sheet = paste0(nombre_sheet, " ")

    # Se emplea la desetacionalización con la opción "outlier = NULL"
    xts_obj = funcion_procesamiento_base(nombre_base, nombre_sheet, ...)

    return(xts_obj)

  })
}
```

4.5.1.2 Función para el procesamiento de todas las bases de datos vía la base de datos `input_web_scraping`

```
# Función para el procesamiento de todos los datos
procesamiento_datos_3PRF = function(input_3PRF_procesamiento_bases, year_filter, name_excel, dir_salida){

  # Iteración a traves de la base que contiene la información de las variables que fueron descargadas
```

```r
for (i in 1:nrow(input_3PRF_procesamiento_bases)){

  # Condicional para crear los objetos xts por cada base procesada (i.e. cada base que se procesa, genera un objeto xts dis

  if(input_3PRF_procesamiento_bases[i,]$Fuente == "DANE"){

    # Función para generar objeto xts de las bases del DANE
    xts_obj = funcion_try_catch(funcion_procesamiento_base = procesamiento_DANE,
                                nombre_base = input_3PRF_procesamiento_bases[i,]$nombre_base,
                                nombre_sheet = input_3PRF_procesamiento_bases[i,]$nombre_sheet,
                                palabra_identificacion_tabla = input_3PRF_procesamiento_bases[i,]$palabra_identificacion_ta
                                nombre_variable = input_3PRF_procesamiento_bases[i,]$nombre_variable)


  }else if (input_3PRF_procesamiento_bases[i,]$Fuente == "CAMACOL"){

    # Función para generar objeto xts de las bases de Camacol
    xts_obj = funcion_try_catch(funcion_procesamiento_base = procesamiento_Camacol,
                                nombre_base = input_3PRF_procesamiento_bases[i,]$nombre_base,
                                nombre_sheet = input_3PRF_procesamiento_bases[i,]$nombre_sheet,
                                palabra_identificacion_tabla = input_3PRF_procesamiento_bases[i,]$palabra_identificacion_ta
                                nombre_variable = input_3PRF_procesamiento_bases[i,]$nombre_variable)

  }else if(input_3PRF_procesamiento_bases[i,]$Fuente == "AEROCIVIL"){

    # Función para generar objeto xts de las bases de la Aerocivil
    xts_obj = funcion_try_catch(funcion_procesamiento_base = procesamiento_Aerocivil,
                                nombre_base = input_3PRF_procesamiento_bases[i,]$nombre_base,
                                nombre_sheet = input_3PRF_procesamiento_bases[i,]$nombre_sheet,
                                palabra_identificacion_tabla = input_3PRF_procesamiento_bases[i,]$palabra_identificacion_ta
                                nombre_variable = input_3PRF_procesamiento_bases[i,]$nombre_variable)

  }else if(input_3PRF_procesamiento_bases[i,]$Fuente == "FNC"){

    # Función para generar objeto xts de las bases de la FNC
    xts_obj = funcion_try_catch(funcion_procesamiento_base = procesamiento_FNC,
                                nombre_base = input_3PRF_procesamiento_bases[i,]$nombre_base,
                                nombre_sheet = input_3PRF_procesamiento_bases[i,]$nombre_sheet,
                                palabra_identificacion_tabla = input_3PRF_procesamiento_bases[i,]$palabra_identificacion_ta
                                nombre_variable = input_3PRF_procesamiento_bases[i,]$nombre_variable)

  }else if(input_3PRF_procesamiento_bases[i,]$Fuente == "DIAN"){

    # Lista de archivos en el directorio
    nombres_bases_de_datos_descargadas = list.files()

    # Uso de la función "str_detect" del paquete "stringr" para detectar el archivo descargado de la DIAN con la palabra "r
    nombre_base_DIAN = nombres_bases_de_datos_descargadas[str_detect(nombres_bases_de_datos_descargadas, "recaudo-mensual")]

    # Función para generar objeto xts de las bases de la DIAN
    xts_obj = funcion_try_catch(funcion_procesamiento_base = procesamiento_DIAN,
                                nombre_base = nombre_base_DIAN,
                                nombre_sheet = input_3PRF_procesamiento_bases[i,]$nombre_sheet,
                                palabra_identificacion_tabla = input_3PRF_procesamiento_bases[i,]$palabra_identificacion_ta
                                nombre_variable = input_3PRF_procesamiento_bases[i,]$nombre_variable)

  }else if(input_3PRF_procesamiento_bases[i,]$Fuente == "BANREP"){

    # Función para generar objeto xts de las bases de la BANREP
    xts_obj = funcion_try_catch(funcion_procesamiento_base = procesamiento_Banrep,
                                nombre_base = input_3PRF_procesamiento_bases[i,]$nombre_base,
                                nombre_sheet = input_3PRF_procesamiento_bases[i,]$nombre_sheet,
                                palabra_identificacion_tabla = input_3PRF_procesamiento_bases[i,]$palabra_identificacion_ta
                                nombre_variable = input_3PRF_procesamiento_bases[i,]$nombre_variable)

  }
```

```
    # Condicional para generar la "matriz" de objetos xts

    # Si es la primer vez que se itera se crea el objeto "xts_matrix" que es la matriz que almacena todas las series de tiempo
    if(i == 1){

      # Inicalización del objeto "xts_matrix" en la primera iteración
      xts_matrix = xts_obj

    }else{

      # En cada iteración se llenar la matriz de objetos xts con un nueva columna ("serie" para cada variable)
      xts_matrix = merge.xts(xts_matrix, xts_obj)

    }

  }

  # Transformación de las variables para que tengan las mismas unidades que las variables que se encuentran en la base origina

  ## Transformación de las variables de importaciones y exportaciones para que tengan las mismas unidades que las variables a

  # Vector con el nombre de las variables para transformar
  variables_para_transformar = c("impo_total_usd",
                                 "impo_consumo",
                                 "impo_intermedios",
                                 "impo_bienescapital",
                                 "expo_total_usd",
                                 "expo_tradicionales",
                                 "expo_no_tradicionales")

  # Transformación de las variables de importaciones y exportaciones para que su valor esté dividido en 1000
  xts_matrix[, variables_para_transformar] = apply(xts_matrix[, variables_para_transformar], 2, function(x) x/1000)

  # Se filtra las series, para que solo contengan información desde "year_filter" en adelante
  xts_filtered = xts_matrix[lubridate::year(index(xts_matrix)) >= year_filter]

  # Se estrae las fechas del objeto "xts_filtered"
  Fecha = index(xts_filtered)

  # Se transforma el objeto "xts_filtered" en una data frame
  todos_df_filter = as.data.frame(xts_filtered)

  # Se transforma los rownames del dataframe en la columna de fechas
  todos_df_filter = rownames_to_column(todos_df_filter, var = "Fecha")

  # Se guarda dicha columna de fechas con los valores que vienen del index de "xts_filtered"
  todos_df_filter = todos_df_filter %>%
    mutate(Fecha = Fecha)

  # Se específica el directorio de salida donde se almacenará la base de datos final
  setwd(dir_salida)

  # Se exporta el dataframe "todos_df_filter como un archivo Excel
  write.xlsx(todos_df_filter, name_excel)

  # Se retorna el objeto "xts_filtered"
  return(xts_filtered)

}
```

4.5.2 Función para actualizar la base de datos del 3PRF

```
# Función para actualizar la base de datos del 3PRF
actualizacion_base_3PRF = function(base_original, base_actualizada_xts, year_filter, name_excel, start_fecha, dir_salida){
```

```r
# Parte del código que prepara las base de datos antes del proceso de actualización de la base del 3PRF

## "base_original_xts"

# Fecha de inicio de la base de datos original ("base_original") del 3PRF
fecha_inicio = ymd(start_fecha)

# Genera la variable que contiente todas las fechas de la base de datos original ("base_original") del 3PRF
fechas = as.yearmon(seq(fecha_inicio, by = "1 month", length.out = nrow(base_original)))

# Modifica "base_original" para que incluya las fechas de la base de datos
base_original = base_original %>%
  mutate(Fecha = fechas)

# Transforma "base_original" (base de datos que contiene las series originales del 3PRF antes de ser actualizadas) en un ob
base_original_xts = xts(base_original[, -1], order.by = base_original$Fecha)

# Vector con los nombres de las variables que se encuentran en la base de datos con las series originales ("base_original_x
names_base_original = names(base_original_xts)

## "base_original_xts"

# Vector con los nombres de las variables que se encuentran en la base de datos con las series actualizadas ("base_actualiz
names_base_actualizada = names(base_actualizada_xts)


# Parte del código de la función que actualiza las bases de datos

# Loop1: Iteración a través de las variables de la base_original_xts del 3PRF
for(i_orig in 1:length(names_base_original)){

  # Vector lógico que permite detectar en que parte de la "base_actualizada_xts" se encuentra la variable de la "base_origi
  # Lo que se está haciendo es una comparación de nombres de las dos bases ("base_actualizada_xts" y "base_original_xts") y
  logical_vector = names_base_original[i_orig] == names_base_actualizada

  # Condición para deterctar si es necesario actualizar los datos de la variable o no.
  ## sum(logical_vector) == 1: Indica que sí, dado que se descargarón datos nuevos
  ## sum(logical_vector) == 0: Indica que no, dado que no se descargarón datos nuevos
  if (sum(logical_vector) == 1){

    # Vector que va almacenar los datos de la variable. Si no se han actualiado deja los de la variable de la base original
    new_vect = c()

    # Índice que indica en que posición de la "base_actualizada_xts" se encuentra la variable de la "base_original_xts"
    i_actual = which(logical_vector)

    # Genero el objeto xts que incluye las dos series de tiempo de la variable de interés, i.e., la anterior ("base_origina
    xts_merge = merge.xts(base_original_xts[, i_orig], base_actualizada_xts[, i_actual])

    # Extraigo el ínidice temporal del objeto "xts_merge" que contiene las dos series de tiempo de la variable de ineterés
    xts_merge_index = index(xts_merge)

    # Extaigo el número de observacione que se encuentran en el objeto "xts_merge"
    xts_merge_nrow = dim(xts_merge)[1]

    # Lleno el nuevo vector que va a contener la información de la variable de interés en la base actualizada
    for (i_merge in 1:xts_merge_nrow){

      # Si la serie actualizada ("base_actualizada_xts") de la variable tiene NA, entonces se dejan los valores de la serie
      if (is.na(xts_merge[, 2][i_merge])){

        # Se llena con los valores de la serie original ("base_original_xts") de la variable
        new_vect[i_merge] = xts_merge[, 1][i_merge]

      }else{
```

```
          # De lo contrario, si la serie actualizada no tiene NA, entonces se llena con los valores de la serie actualizada (
          new_vect[i_merge] = xts_merge[, 2][i_merge]
        }

      }

      # Se transforma el vector que se acaba de llenar en un objeto xts
      new_xts = xts(new_vect, order.by = xts_merge_index)

    }else{

      # En caso de que no haya un versión actualizada de la variable (i.e. la varible no se encuentre en "base_actualizada_xts
      new_xts = base_original_xts[, i_orig]

    }

    # Condicional para generar la "matriz" de objetos xts

    # Si es la primer vez que se itera se crea el objeto "xts_matrix" que es la matriz que almacena todas las series de tiempo
    if(i_orig == 1){

      # Inicalización del objeto "xts_matrix" en la primera iteración
      xts_matrix = new_xts

    }else{

      # En cada iteración se llenar la matriz de objetos xts con un nueva columna ("serie" para cada variable)
      xts_matrix = merge.xts(xts_matrix, new_xts)

    }

  }

  # Se renombran las variables de "xts_matrix" para que los nombres de las series correspondan a los mismos nombre de las vari
  names(xts_matrix) = names_base_original

  # Se filtra las series, para que solo contengan información desde "year_filter" en adelante
  xts_filtered = xts_matrix[lubridate::year(index(xts_matrix)) >= year_filter]

  # Se estrae las fechas del objeto "xts_filtered"
  Fecha = index(xts_filtered)

  # Se transforma el objeto "xts_filtered" en una data frame
  todos_df_filter = as.data.frame(xts_filtered)

  # Se transforma los rownames del dataframe en la columna de fechas
  todos_df_filter = rownames_to_column(todos_df_filter, var = "Fecha")

  # Se guarda dicha columna de fechas con los valores que vienen del index de "xts_filtered"
  todos_df_filter = todos_df_filter %>%
    mutate(Fecha = Fecha)

  # Se específica el directorio de salida donde se almacenará la base de datos final
  setwd(dir_salida)

  # Se exporta el dataframe "todos_df_filter como un archivo Excel
  write.xlsx(todos_df_filter, name_excel)

  # Se retorna el objeto "xts_filtered"
  return(xts_filtered)

}
```

5 Desestacionalización de las variables del 3PRF

6 Modelo 3PRF

7 Manual del operario: Ejecución del modelo 3PRF por parte del operario

7.1 Procedimiento para operar el modelo 3PRF

7.2 Script del operario del modelo 3PRF

```
# Código proceso de Web Scraping completo en R

# 1. Preliminares ------------------------------------------------------

# Limpieza del entorno de trabajo
rm(list = ls())

# Directorio de trabajo

## Scripts
scripts = "C:/Users/germa/Desktop/UNAL/BID/Proyecto_web_scraping_personal/web_scraping_3PRF/V2/scripts"

## Bases de datos

# Entrada
input = "C:/Users/germa/Desktop/UNAL/BID/Proyecto_web_scraping_personal/web_scraping_3PRF/V2/bases_datos/input_web_scraping"

# Salida
bases_descargadas = "C:/Users/germa/Desktop/UNAL/BID/Proyecto_web_scraping_personal/web_scraping_3PRF/V2/bases_datos/output_w
bases_finales = "C:/Users/germa/Desktop/UNAL/BID/Proyecto_web_scraping_personal/web_scraping_3PRF/V2/bases_datos/output_web_s

# Imporación de las funciones para realizar el proceso de Research
setwd(scripts)

# Se traen de los scripts las funciones para el proceso de Web Scraping

## Script con las funciones de descarga
source("web_scraping_R_funciones_descarga.R")

## Script con las funciones para el procesamiento de las bases de datos
source("web_scraping_R_funciones_procesamiento_bases.R")

# 2. Importación de datos ----------------------------------------------
setwd(input)

# Excel con el input necesario para
input_3PRF_descarga = read_xlsx("input_web_scraping.xlsx", sheet = "descarga")

# Excel con el input necesario para
input_3PRF_procesamiento_bases = read_xlsx("input_web_scraping.xlsx", sheet = "procesamiento_bases")

# Excel con la base original del 3PRF antes de la actualización de los datos
base_original = read.xlsx("nuevo_variables 3PRF.xlsx", sheet = "Variables (Base)", startRow = 5)

# 3. Descarga de las bases de datos vía Web Scraping -------------------------------------------------
setwd(bases_descargadas)

# Descarga de todas las bases de datos
# descarga_datos_3PRF(input_3PRF_descarga, tmp_sleep = 30)

# 4. Procesamiento de las bases de datos descargadas por Web Scraping ----------------------------------------------

# Función que realiza el procesamiento de todas las bases de datos descargadas
```

```r
base_actualizada_xts = procesamiento_datos_3PRF(input_3PRF_procesamiento_bases,
                                                year_filter = 2000,
                                                name_excel = "base_variables_descargadas_3PRF.xlsx",
                                                dir_salida = bases_finales); base_actualizada_xts

# Función que realiza la actualización de la base del 3PRF
base_3PRF_final_xts = actualizacion_base_3PRF(base_original,
                                               base_actualizada_xts,
                                               year_filter = 2000,
                                               name_excel = "base_3PRF_actualizada.xlsx",
                                               start_fecha = "2000-01-01",
                                               dir_salida = bases_finales); base_3PRF_final_xts
```

8 Apéndice

Bigliografía

Wickham, Hadley. 2014. «Tidy Data». *Journal of Statistical Software* 59 (10). https://doi.org/10.18637/jss.v059.i10.

———. 2024. «rvest: Easily Harvest (Scrape) Web Pages». https://CRAN.R-project.org/package=rvest.