Manual para el procesamiento y análisis de información de encuestas con técnicas econométricas para estudios SEB

Fecha: Abril de 2026

Lugar: Lima, Perú

Contenido

Introducción

Los estudios de beneficios socioeconómicos (SEB) de los servicios climáticos buscan demostrar que el verdadero valor de estos servicios radica en cómo la información que proveen es utilizada por los usuarios para tomar decisiones que impactan positivamente en los resultados sociales y económicos. Estos estudios son esenciales para evidenciar la relevancia de los servicios de tiempo, agua y clima (TAC) en la gestión del riesgo y la mejora de la resiliencia frente a fenómenos hidrometeorológicos.

En el marco del Proyecto ENANDES+, orientado a fortalecer la resiliencia climática en los países andinos (Argentina, Bolivia, Ecuador y Perú), se promueve el desarrollo de capacidades técnicas que permitan mejorar la evaluación de los beneficios socioeconómicos de los servicios climáticos. Para ello, se impulsa el uso de herramientas estadísticas y econométricas que faciliten el procesamiento, análisis y visualización de datos de encuestas, contribuyendo a la generación de evidencia para la toma de decisiones informadas.

RStudio, entorno de desarrollo del software libre R, constituye una herramienta clave para este propósito. R permite realizar análisis estadísticos, programar y generar gráficos de manera flexible y reproducible, lo que lo convierte en una plataforma idónea para aplicar técnicas econométricas y procesar información de encuestas vinculadas a los estudios SEB.

Este manual ha sido elaborado para guiar a los usuarios en el uso de R y RStudio, desde su instalación y configuración inicial hasta la aplicación práctica de métodos estadísticos y econométricos. Con ello, se busca fortalecer las capacidades técnicas de los profesionales y socios regionales vinculados al proyecto ENANDES+, apoyando la misión del SENAMHI – Perú y la Organización Meteorológica Mundial (OMM) en la generación de conocimiento y en la promoción de una gestión climática más informada, eficiente y resiliente.

1. Introducción al análisis de bases de datos

El análisis de datos es un procedimiento mediante el cual se examina un conjunto de información con el propósito de extraer conclusiones, ampliar el conocimiento y respaldar la toma de decisiones. En el ámbito de las instituciones públicas, este proceso es fundamental para diseñar, implementar y evaluar políticas y programas basadas en evidencia, optimizar la asignación de recursos, mejorar la focalización de programas sociales, medir resultados e identificar brechas territoriales o poblacionales.

1.1. Tipos de análisis de datos

El análisis de datos puede tener distintos enfoques en función del objetivo del estudio o del tipo de política o programa que se quiera implementar. Cada enfoque responde a preguntas específicas y utiliza herramientas metodológicas diferenciadas, que van desde técnicas descriptivas básicas hasta modelos econométricos y predictivos más avanzados. A continuación, se presentan los principales tipos de análisis, junto con las

preguntas que buscan responder y las técnicas metodológicas que se emplean en cada caso.

Tabla N° 1. Tipos de análisis de datos

Tipos	Definición	Preguntas	Técnicas
Descriptivo	Resume y organiza los datos.	¿Qué ha pasado?	Tablas de frecuencia, medidas de tendencia central y dispersión.
Diagnóstico	Busca causas o factores asociados a un fenómeno.	¿Por qué pasó?	Comparación entre grupos, pruebas estadísticas, correlaciones.
Exploratorio	Descubre información oculta.	¿Qué patrones hay?	Encuestas, entrevistas, grupos focales.
Inferencial	Usa muestras para hacer estimaciones sobre la población.	¿Se puede generalizar?	Intervalos de confianza, modelos probabilísticos.
Predictivo	Pronostica tendencias o comportamientos.	¿Qué pasará?	Series de tiempo, machine learning, modelos predictivos.

Elaboración: propia.

1.2. Indicadores de calidad de los datos

Además de definir el tipo de análisis que se aplicará, es fundamental asegurar la calidad de los datos, ya que de ello depende la validez y confiabilidad de los resultados. Diversos organismos han establecido criterios y estándares para evaluar la calidad de la información. En este marco, el Banco Mundial[1] propone seis dimensiones clave que permiten validar y garantizar la calidad de los datos utilizados en el análisis.

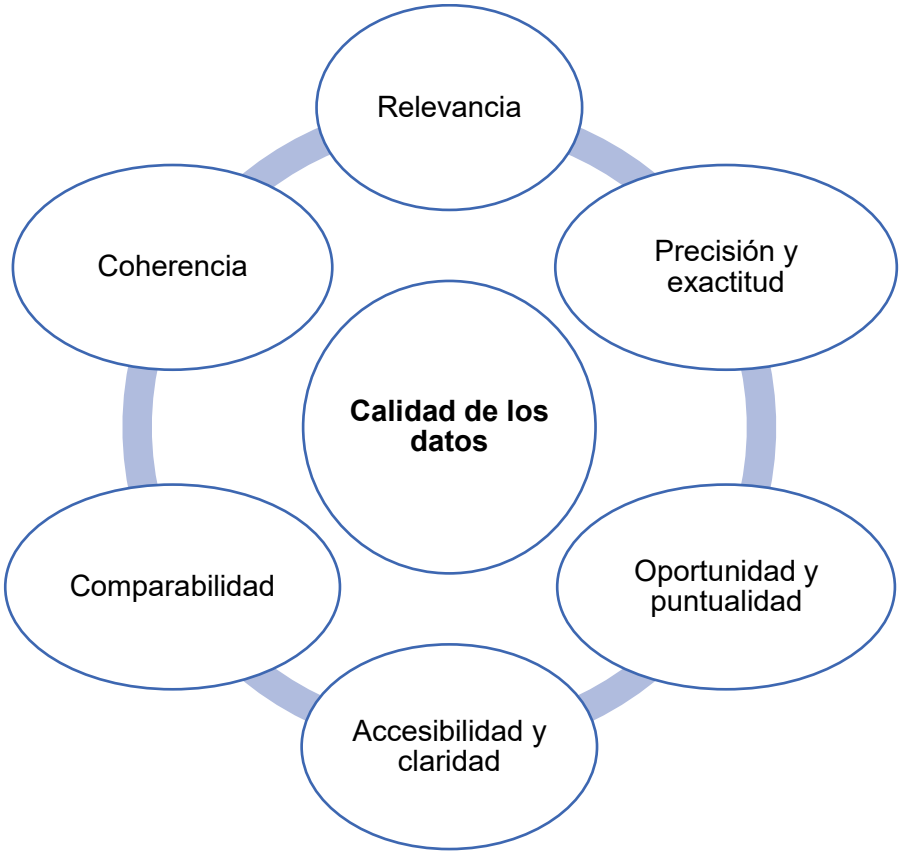
- **Relevancia**: Indica el grado en que los datos satisfacen las necesidades actuales y potenciales de los usuarios. Esto se debe medir mediante la consulta directa a los usuarios.
- **Precisión y exactitud**: Indica la proximidad entre el valor estimado y el valor real. También se puede expresar en términos de errores de muestreo, de cobertura, de no respuesta o en el mismo procesamiento.

[1] Banco Mundial: https://www.bancomundial.org/es/data/opendatatoolkit/supply

- **Oportunidad y puntualidad**: Indica que tan rápido se publican los datos en relación con lo que miden y en función de fechas anunciadas en algún calendario oficial.
- **Accesibilidad y claridad**: Indica la facilidad con la que los usuarios pueden acceder a losa datos y el grado de en el que se explican los datos.
- **Comparabilidad**: Indica el grado en que los datos se puedan comparar en el ámbito temporal, geográfico y no geográfico (otros dominios de interés).
- **Coherencia:** Indica el grado en que los datos se ajustan a las definiciones o metodologías reconocidas. Además, se refiere a la integración entre estadísticas diferentes.

En conclusión, la calidad de los datos constituye el pilar sobre el cual se construye todo análisis estadístico o econométrico. Sin información pertinente, precisa y coherente, incluso las técnicas más sofisticadas pueden conducir a conclusiones equivocadas. Por ello, incorporar criterios de evaluación de calidad desde el inicio del proceso analítico no solo mejora la robustez de los resultados, sino que también fortalece la transparencia, la rendición de cuentas y la efectividad de las decisiones públicas basadas en evidencia.

Figura N° 1. Normas Generales de Calidad



Fuente: Banco Mundial.

1.3. Procesamiento de datos

El procesamiento de datos abarca la preparación de la información desde su recolección hasta su análisis final. Este proceso incluye la limpieza de datos, donde se identifican y corrigen errores, inconsistencias y valores atípicos; la transformación, que permite adecuar las variables o la estructura de la base según los requerimientos metodológicos; la codificación, fundamental para convertir información categórica en formatos cuantificables y analizables; y, finalmente, el análisis, etapa en la que se aplican técnicas estadísticas y econométricas para obtener resultados sólidos y útiles para la toma de decisiones.

- **Limpieza[2]:** Este proceso se identifican la información incorrecta, incompletas, inexacta o atípica de los datos. Estos datos "sucios" pueden ser reemplazados, modificados o eliminados. En este proceso se pueden encontrar:
  - Valores duplicados: registros repetidos que pueden generar sesgos en los resultados.
  - Valores atípicos (outliers): observaciones poco comunes que presentan valores extremos y pueden distorsionar las estimaciones.
  - Valores faltantes (missing values): datos que no fueron registrados o no pudieron recopilarse.
  - Valores inconsistentes: información que no sigue una lógica interna (por ejemplo, edades negativas o fechas imposibles).
  - Registros incompletos: observaciones con variables clave sin información suficiente.
  - Respuestas no válidas o falsas: casos donde se detectan patrones incoherentes o respuestas intencionalmente incorrectas.

- **Transformación[3]:** Este proceso consiste en modificar la estructura, escala o representación de las variables con el fin de adecuarlas a los requerimientos del análisis estadístico o econométrico. Entre los principales métodos de transformación se encuentran:
  - Normalización de valores, generalmente entre 0 y 1.
  - Estandarización del formato de las variables (entero, decimal, texto, fecha, categórica, etc.).
  - Transformaciones matemáticas, se aplican operaciones como logaritmos, raíces, divisiones, porcentajes, sumas, restas, etc.
  - Creación de nuevas variables derivados de variables existentes.
  - Restructuración de la base, cambiar de formato largo a formato ancho.
  - Unión de bases de datos.

- **Codificación:** Este proceso consiste en organizar y estructurar las variables para facilitar su análisis. Incluye la generación de etiquetas, la estandarización

[2] Joshi, A. P., & Patel, B. V. (2020).

[3] Koukaras, P., & Tjortjis, C. (2025).

de categorías de respuesta y la elaboración de un diccionario de datos que describa el contenido, tipo y significado de cada variable. Este proceso garantiza coherencia y correcta interpretación de la información.

Estos tres procesos son fundamentales para poder realizar un análisis riguroso y confiable. Una adecuada limpieza, transformación y codificación de los datos permite reducir errores, mejorar la consistencia de la información y asegurar que las técnicas estadísticas y econométricas se apliquen correctamente.

2. ¿Qué es R y RStudio?

Es un software libre y de código abierto para el análisis estadístico, la manipulación de datos y la visualización gráfica. Fue creado en 1993 por Robert Gentleman y Ross Ihaka en la Universidad de Auckland (Nueva Zelanda), y desde 1997 se desarrolla de manera colaborativa bajo la coordinación del R Core Team. Está disponible bajo la Licencia Pública General de GNU (GPL), lo que garantiza su libre uso, modificación y distribución.

R combina las características de un software estadístico y un lenguaje de programación. Esto le otorga gran versatilidad para realizar tareas que van desde análisis descriptivos básicos hasta modelamientos econométricos y simulaciones complejas. Se considera un lenguaje interpretado y orientado a objetos, lo que permite escribir instrucciones en secuencia y ejecutarlas de forma inmediata.

RStudio es un entorno de desarrollo integrado (IDE) para trabajar con el lenguaje R. Facilita escribir código, analizar datos, crear gráficos y desarrollar proyectos de manera organizada y eficiente.

2.1. Principales características y ventajas

- ✓ **Código abierto y libre:** permite el acceso gratuito y la mejora continua de la comunidad global.
- ✓ **Extensible mediante paquetes**: el ecosistema de R se organiza en “paquetes” o librerías que amplían sus funciones. Existen más de 15,000 paquetes disponibles en el repositorio oficial CRAN (Comprehensive R Archive Network), alojado en múltiples servidores o *mirrors* en todo el mundo.
- ✓ **Multiplataforma**: puede instalarse y ejecutarse en sistemas Windows, macOS y Linux.
- ✓ **Alta capacidad gráfica**: permite crear visualizaciones personalizadas en formato estático o interactivo.
- ✓ **Reproducibilidad**: los análisis pueden documentarse y replicarse fácilmente, favoreciendo la transparencia científica.
- ✓ **Interoperabilidad**: se integra con múltiples lenguajes y formatos de datos (CSV, Excel, Stata, SPSS, SQL, entre otros).

El sistema base de R incluye paquetes fundamentales como utils, stats, datasets y graphics, así como otros recomendados (boot, class, cluster, entre otros). Para análisis

más especializados, los usuarios pueden instalar librerías adicionales desde CRAN o repositorios personales.

R presenta una serie de características que lo convierten en una herramienta altamente eficiente para el análisis de datos. Sus funcionalidades no solo facilitan el procesamiento y modelamiento estadístico, sino que también fortalecen la calidad, reproducibilidad y alcance de los estudios. A continuación, se detallan las principales ventajas que ofrece este entorno de programación para el trabajo analítico y econométrico.

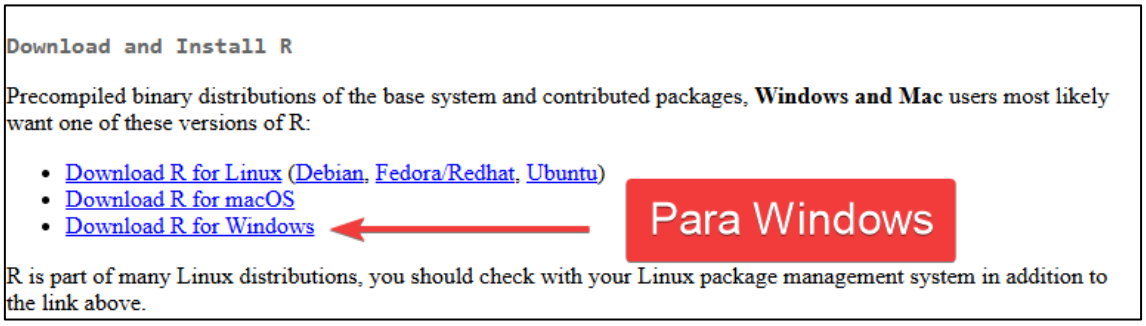
- Sintaxis que permite la replicabilidad, eficiencia y control del análisis de datos.
- Gráficos de calidad profesional con ggplot2 y librerías interactivas.
- Gran variedad de paquetes especializados para el análisis estadístico y econométrico.
- Compatible con otros lenguajes y herramientas (Python, SQL, SAS, CSV, Excel, Stata).
- Es actualizado constantemente.
- Amplia cantidad de recursos de ayuda en línea.

2.2. Instalación de R

Primero se debe instalar R, que es un programa básico con una interfaz simple; y segundo, debe instalar RStudio. En este apartado se muestra como encontrar los archivos ejecutables para la instalación de R y RStudio. Para la instalación de ambos programas, simplemente acepte los valores predeterminados ofrecidos por los programas durante la instalación.

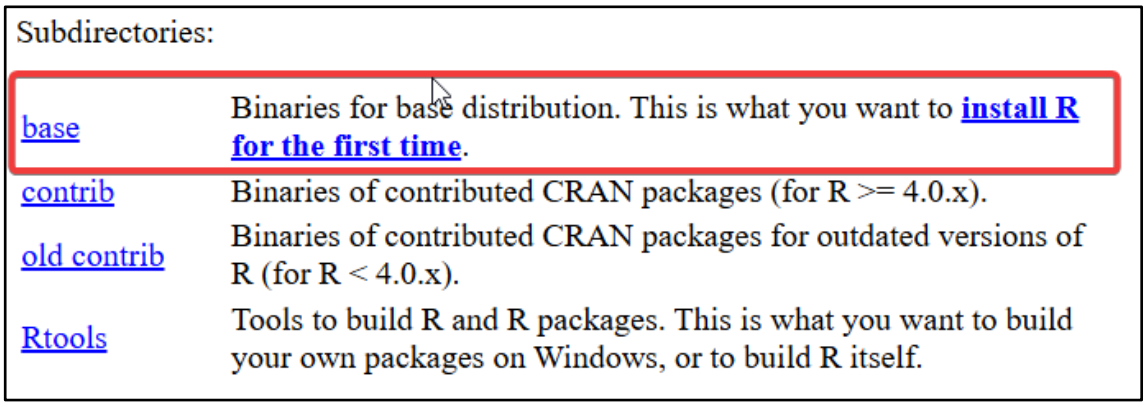
Paso 1: Descargar R

Para instalar el sistema base o entorno R, se deben seguir las instrucciones de la página de CRAN http://cran.r-project.org. Instale la versión correcta de R para el sistema operativo de su computadora.



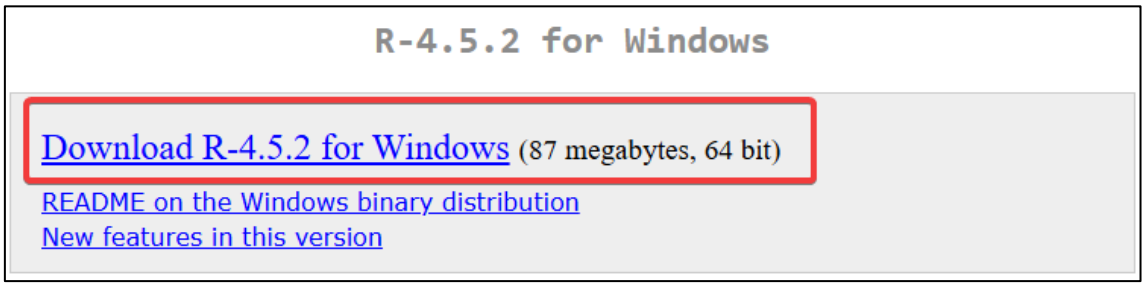
Paso 2: Seleccionar el subdirectorio "base"

Cuando instalamos R por primera vez, se debe seleccionar el subdirectorio llamado "base".



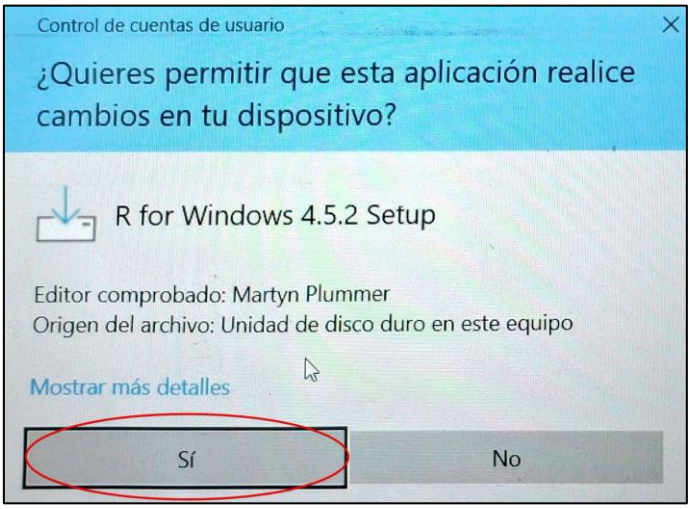
Paso 3: Hacer clic sobre "Download R-4.5.2 for Windows (87 megabytes, 64 bit)"

Es importante mencionar que existen actualizaciones de R, por tanto, es seguro que aparezcan nuevas versiones.

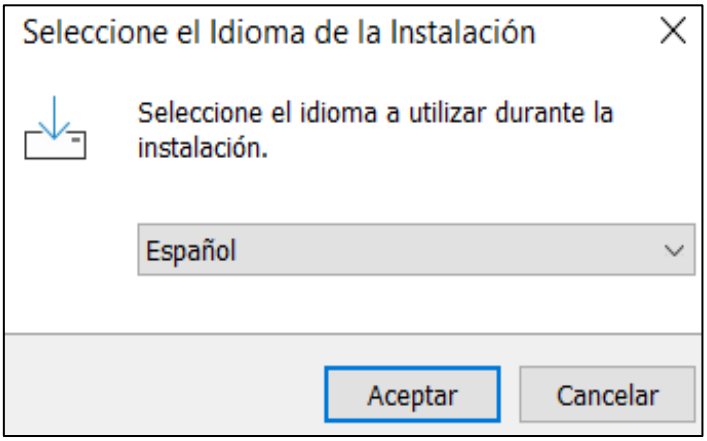


Nota: Revisar en qué carpeta se ha guardado el archivo descargado.

Paso 4: Ejecutar como administrador el archivo descargado

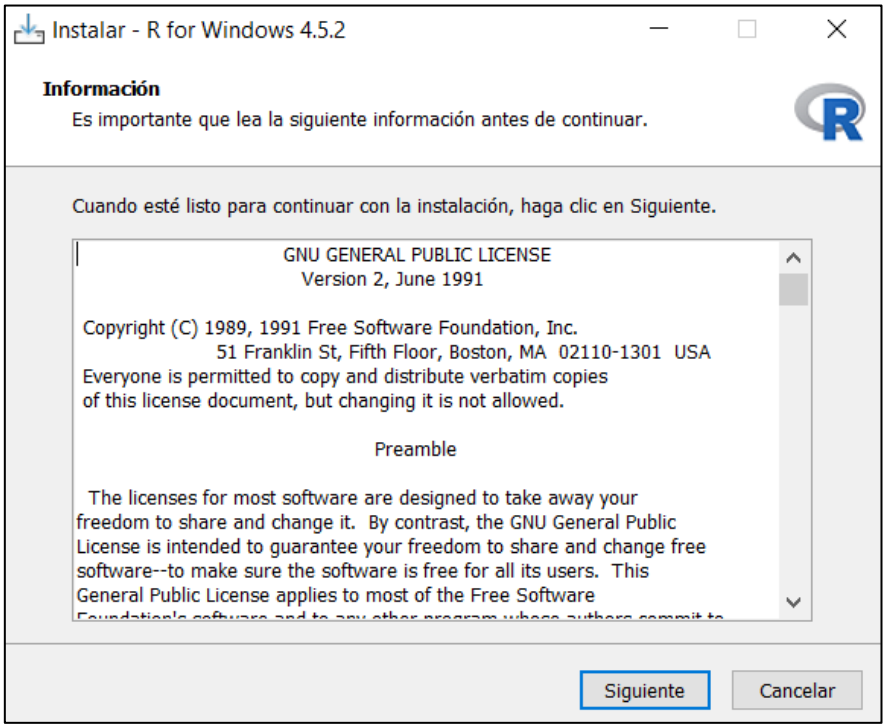


Paso 5: Seleccionar el idioma del sistema

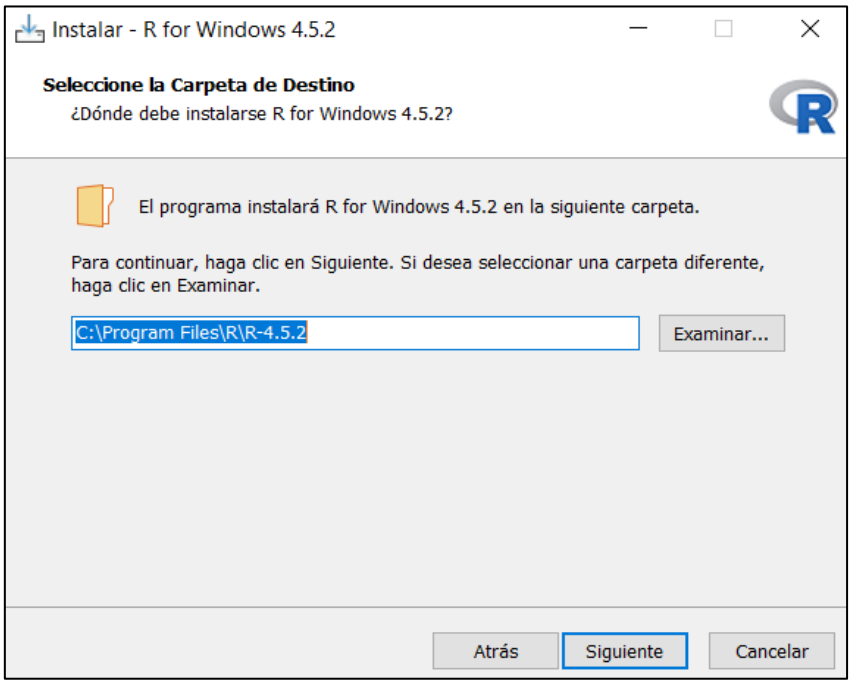


Paso 6: Seguir las instrucciones del Asistente de Instalación de R

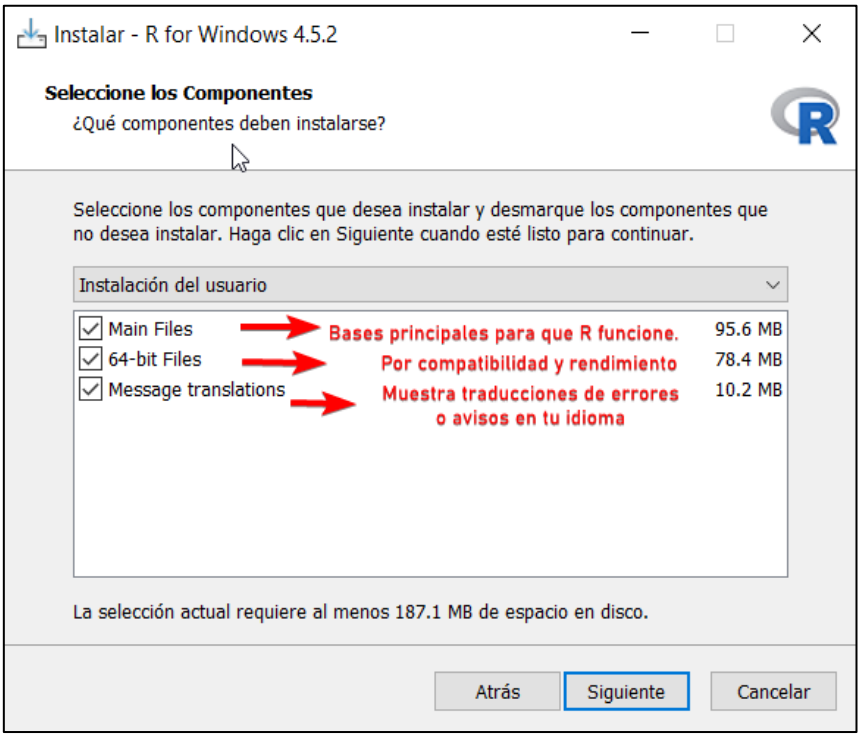
- Leemos la información de instalación y dar clic en “Siguiente”.



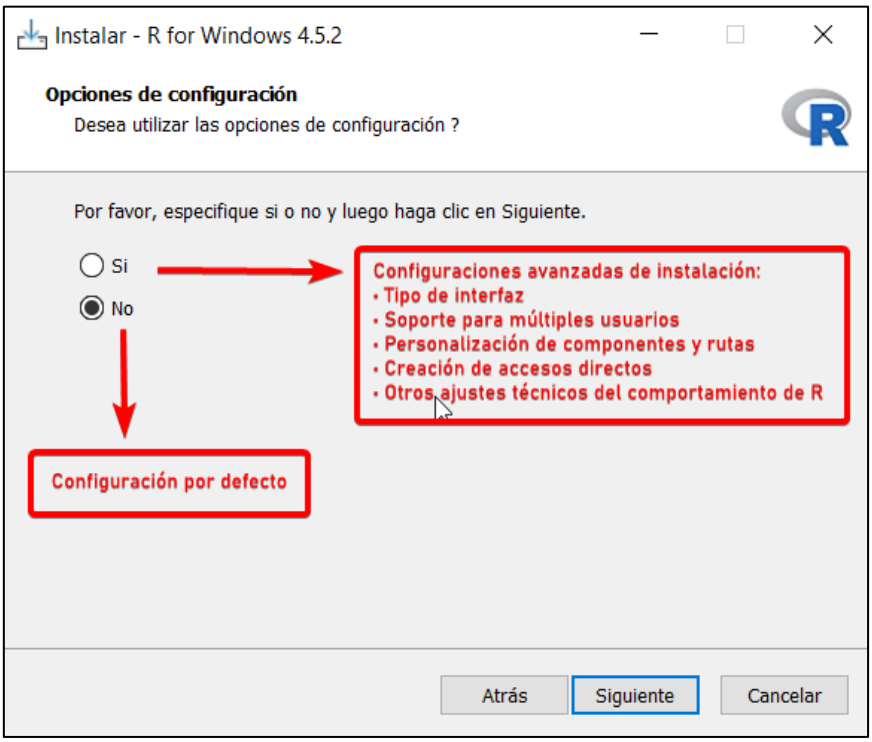
- Seleccionamos la carpeta donde instalaremos R. Se recomienda que sea en el disco C.



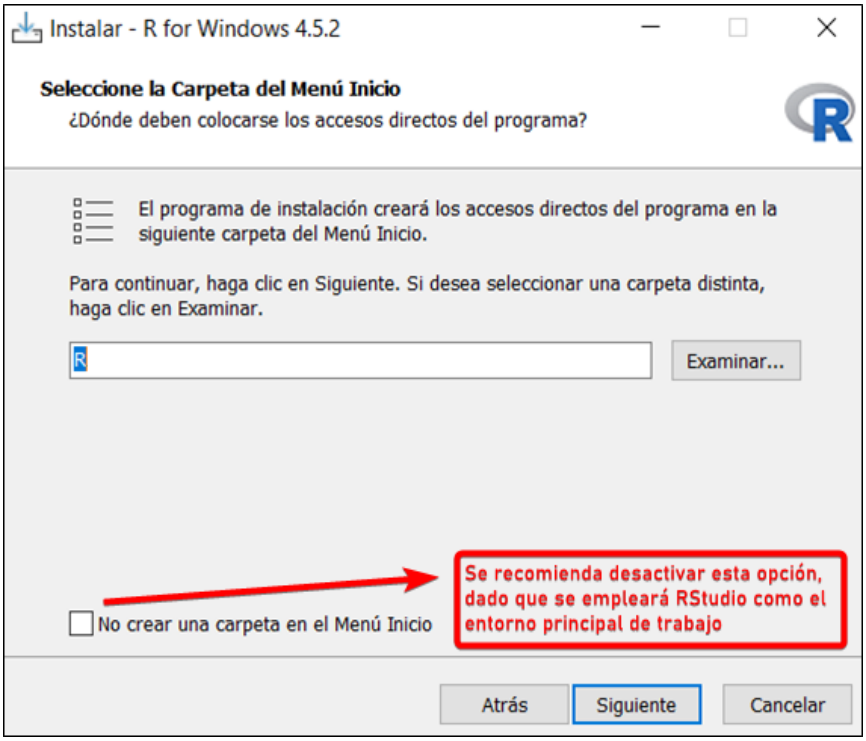
- Seleccionamos los componentes a instalar. Se recomienda seleccionar todos los componentes.



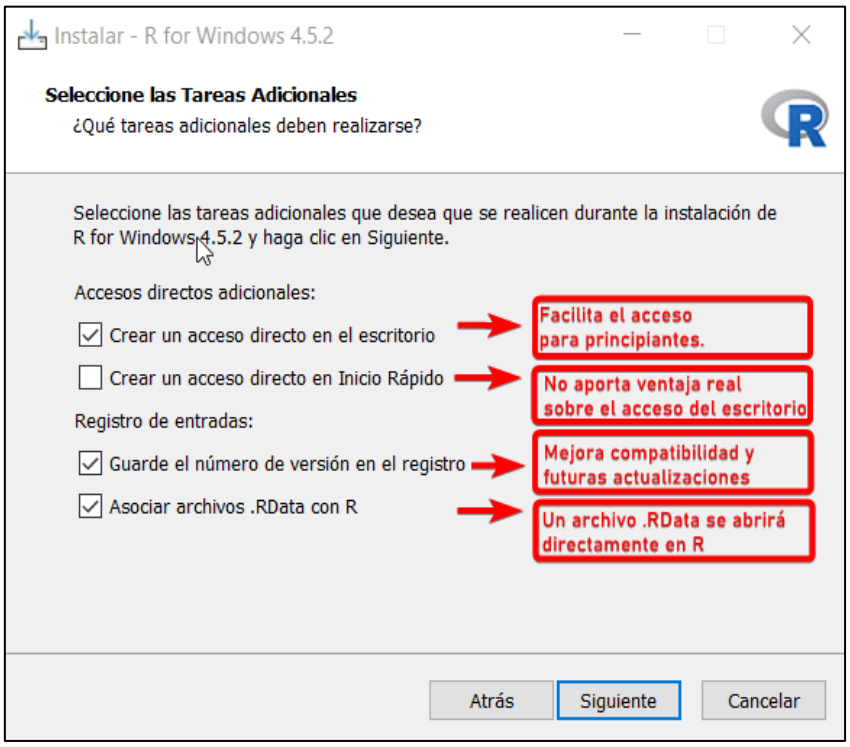
- Especificamos si utilizaremos opciones de configuración



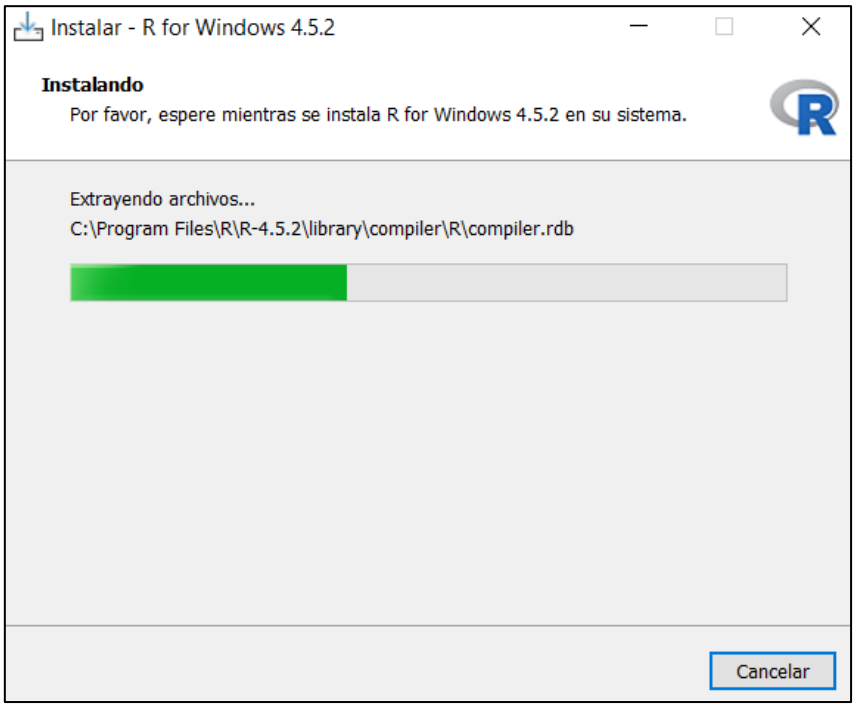
- Seleccionamos dónde se crearán accesos directos al programa



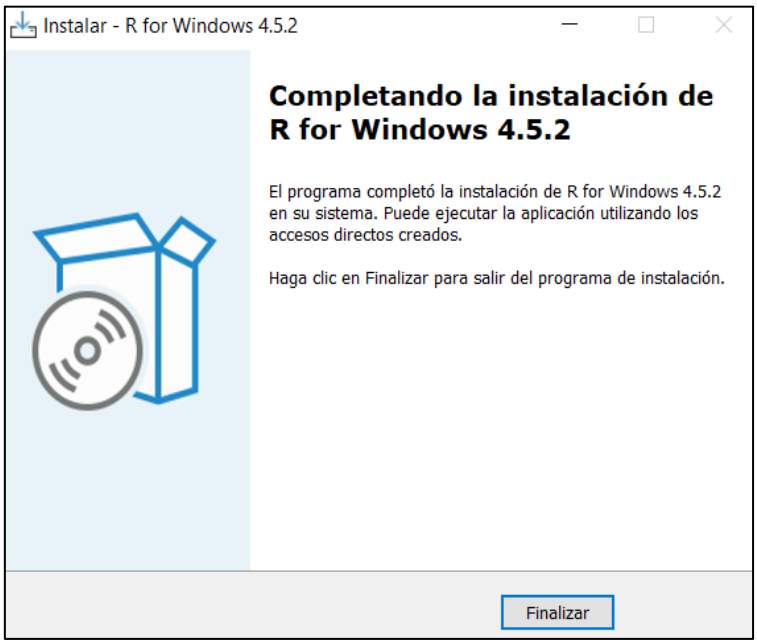
- Seleccionamos tareas adicionales como la de “Crear un ícono en el escritorio”.



- Una vez ejecutadas las acciones anteriores, R se instalará automáticamente.



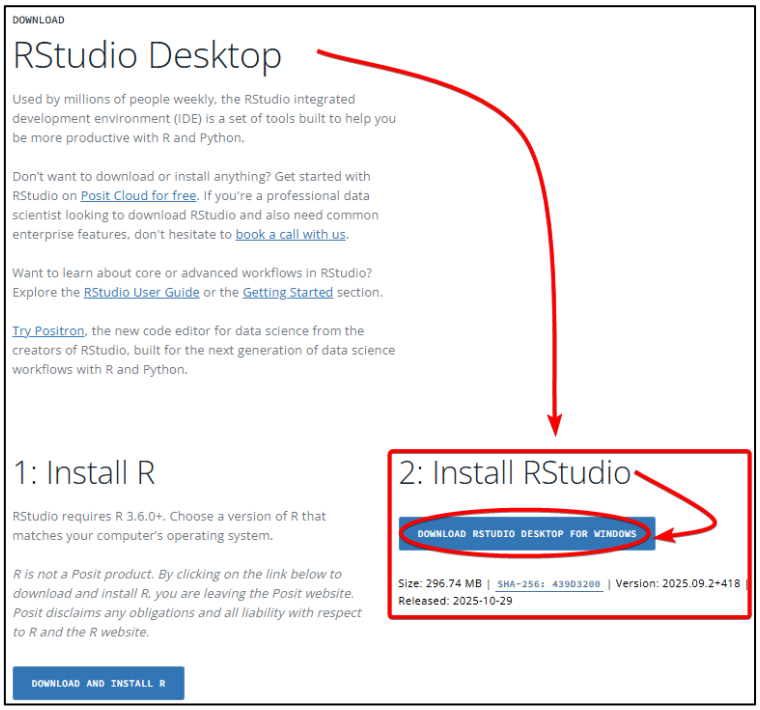
- Para terminar el proceso hacemos clic sobre el botón “Finalizar”.



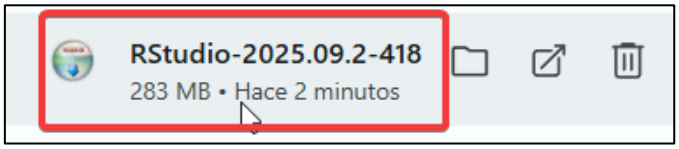
2.3. Instalación de RStudio

Paso 1: Descargar RStudio

Para instalar RStudio vaya a la página https://posit.co/download/rstudio-desktop/. Siga las instrucciones para instalar la versión correcta de RStudio en su computadora.

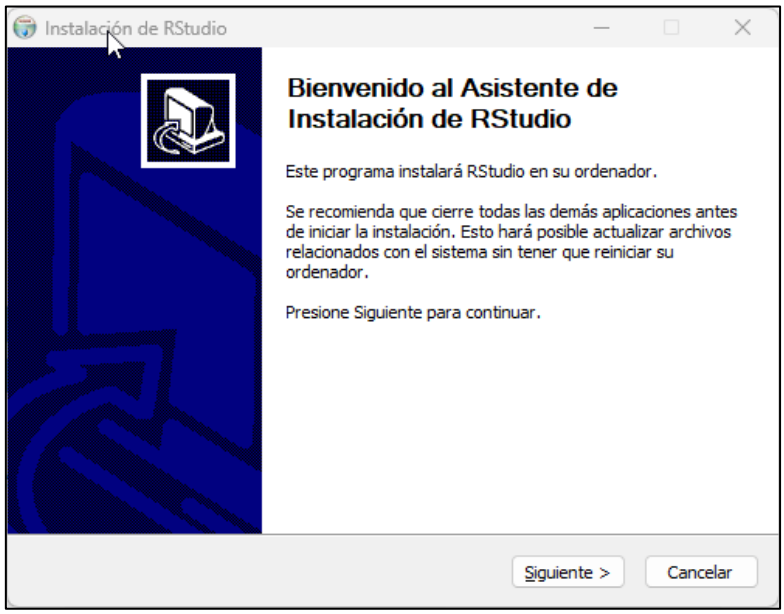


Paso 2: Ejecutar como administrador el archivo descargado

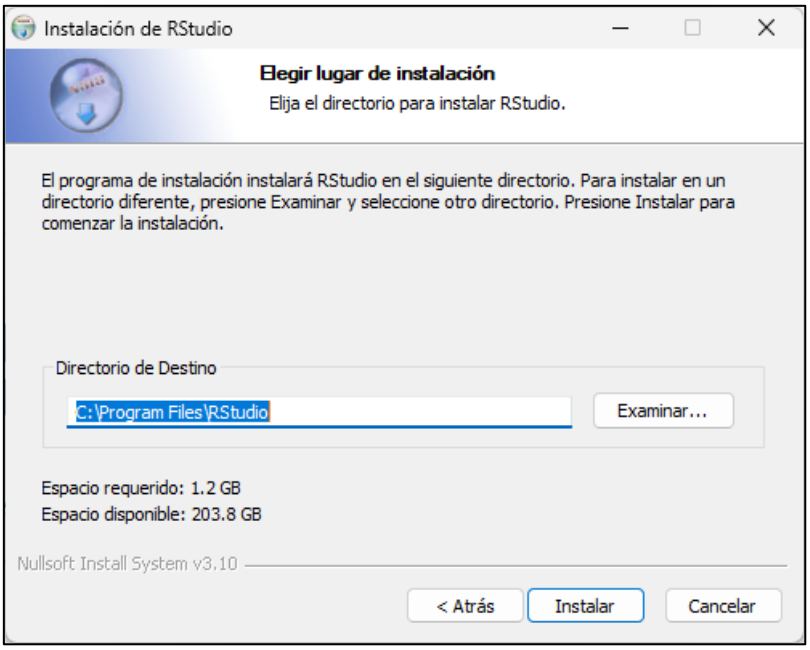


Paso 3: Seguir las instrucciones del Asistente de Instalación de RStudio

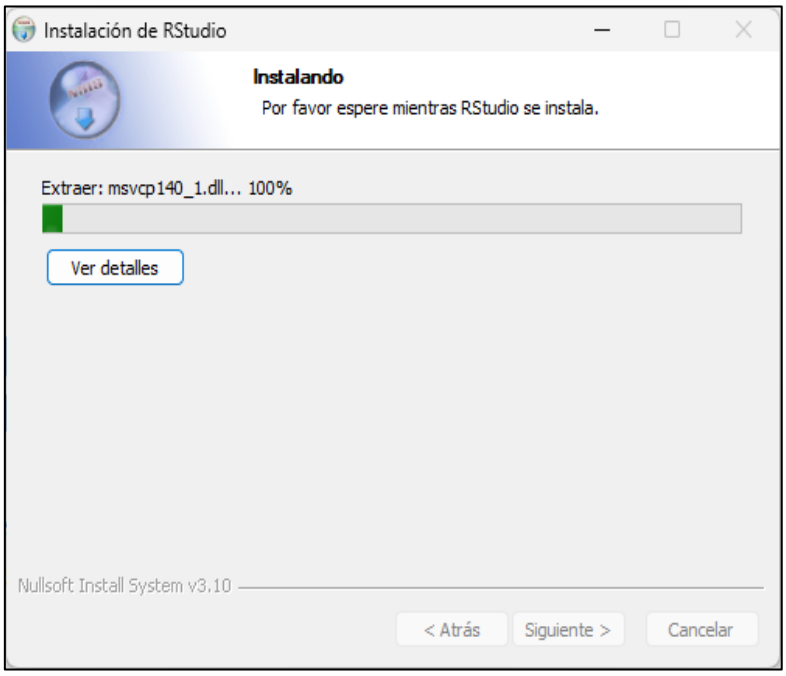
- Leemos la información de instalación y dar clic en “Siguiente”.



- Seleccionamos la carpeta donde instalaremos RStudio.



- Una vez ejecutadas las acciones anteriores, RStudio se instalará automáticamente.



- Para terminar el proceso hacemos clic sobre el botón “Terminar”.

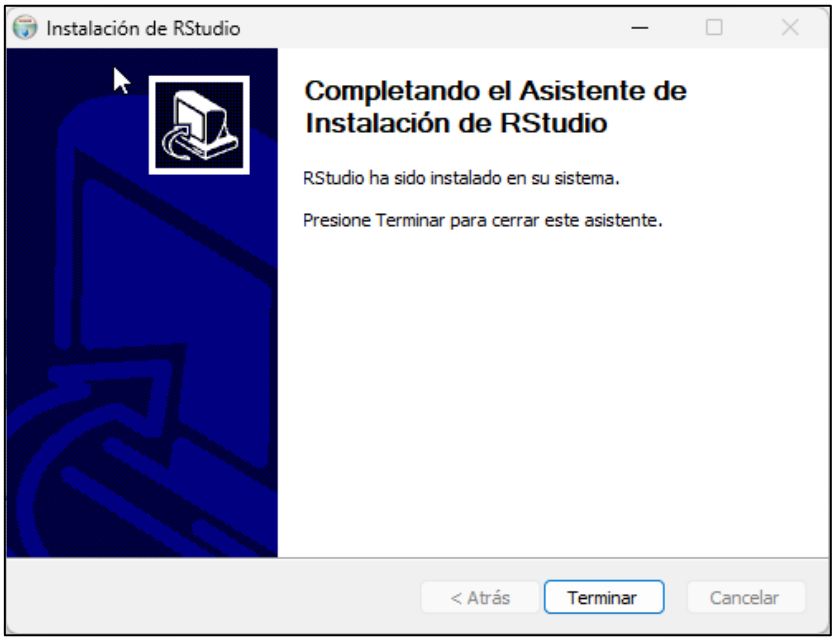


Figura N° 2 Flujograma Proceso de Instalación de R y RStudio

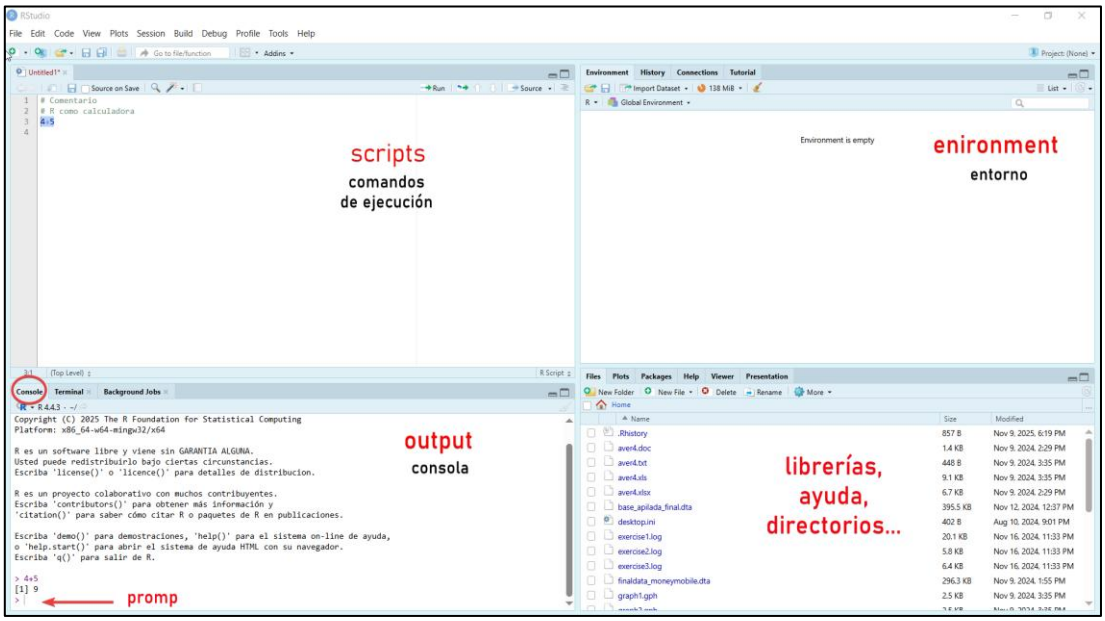
Elaboración: Propia

3. Entorno de trabajo de RStudio

R funciona principalmente a través de una consola de comandos donde el usuario escribe instrucciones que el intérprete ejecuta. Aunque es posible trabajar directamente desde la consola de R, el entorno RStudio ofrece una interfaz más amigable y completa, ya que organiza el flujo de trabajo en cuatro paneles principales:

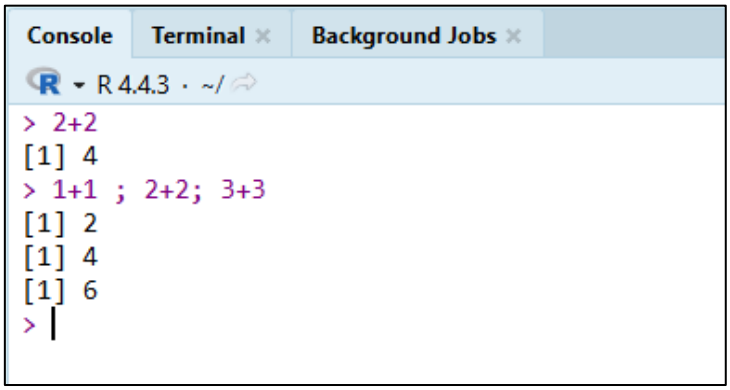
- **Consola:** donde se ejecutan los comandos y se muestran los resultados.
- **Editor de scripts:** para escribir, guardar y ejecutar código de forma ordenada.
- **Environment/History:** muestra los objetos creados y el historial de comandos.
- **Files/Plots/Packages/Help:** permite gestionar archivos, visualizar gráficos, cargar paquetes y acceder a la ayuda.

Esta estructura facilita el análisis de datos, la depuración de código y la gestión de proyectos en R.



3.1. Consola

Por defecto, la consola se ubica en el panel inferior izquierdo. En ella aparece información inicial y, al final, el símbolo “>”, que indica que R está listo para recibir instrucciones. Aquí puedes escribir comandos simples como 2+2 o ejecutar varias instrucciones a la vez, por ejemplo 1+1; 2+2; 3+3. Finalmente, presiona ***Enter*** para que R procese los comandos y muestre los resultados.

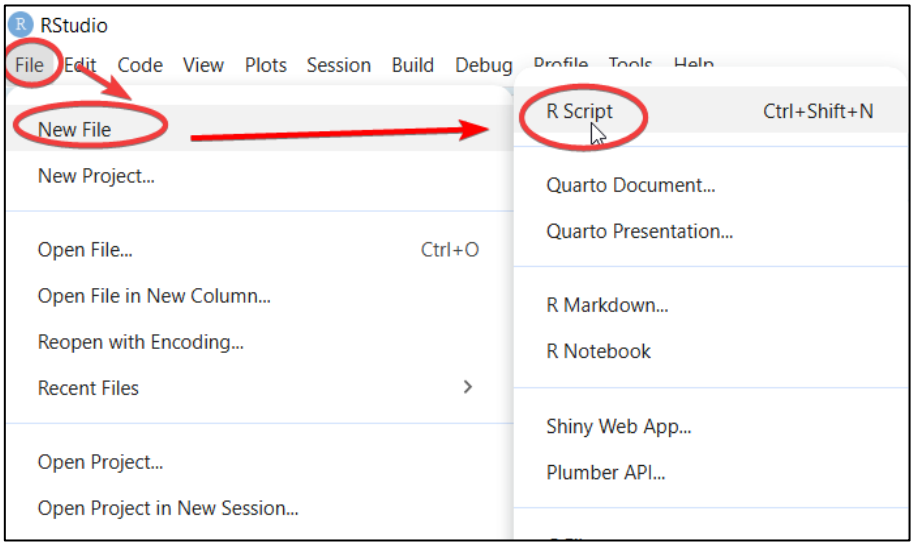


3.2. Scripts

Trabajar únicamente desde la consola puede resultar limitado, ya que las instrucciones deben ejecutarse una por una y no quedan guardadas para su reutilización. Por ello, lo más habitual es trabajar con scripts, es decir, archivos de texto que contienen secuencias de comandos organizados y editables. Estos archivos, con extensión “.R”, permiten escribir, guardar, modificar y ejecutar bloques completos de código, lo que facilita la reproducción de análisis, el orden en el trabajo y la documentación de cada paso realizado.

3.2.1. Crear script

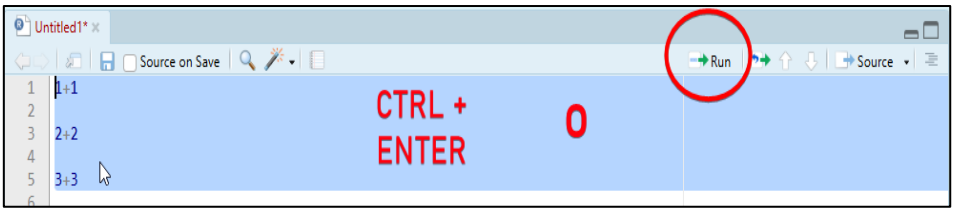
Para crear una script desde RStudio, seleccionamos la siguiente ruta de menús: File > New File > R script



3.2.2. Ejecutar líneas de código

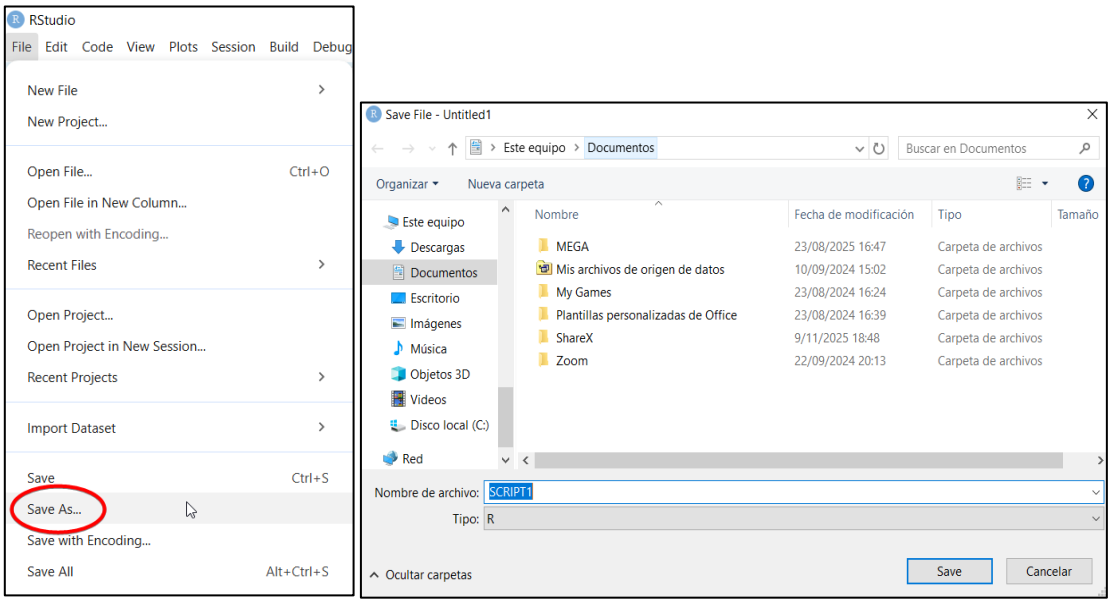
El panel del script se encuentra en la parte superior izquierda de RStudio. En este espacio puedes escribir tus instrucciones de manera ordenada, línea por línea. Cada comando puede ejecutarse individualmente o en bloques seleccionando varias líneas a la vez. Para ejecutar el código, RStudio ofrece varias opciones, lo que facilita trabajar de forma más rápida y eficiente.

Para ejecutar el código existen dos opciones: (1) Dar clic en **Run**, (2) Presionar **CTRL+ENTER**. En ambos casos se recomienda primero seleccionar las líneas de código que se quieren ejecutar.



3.2.3. Guardar script

Para guardar un script, puedes utilizar el menú **File > Save As…** y elegir la carpeta donde deseas almacenar el archivo. Otra opción es hacer clic en el ícono **Guardar** ubicado en la parte superior del panel del script. Esto te permite conservar tu trabajo y retomarlo fácilmente más adelante.



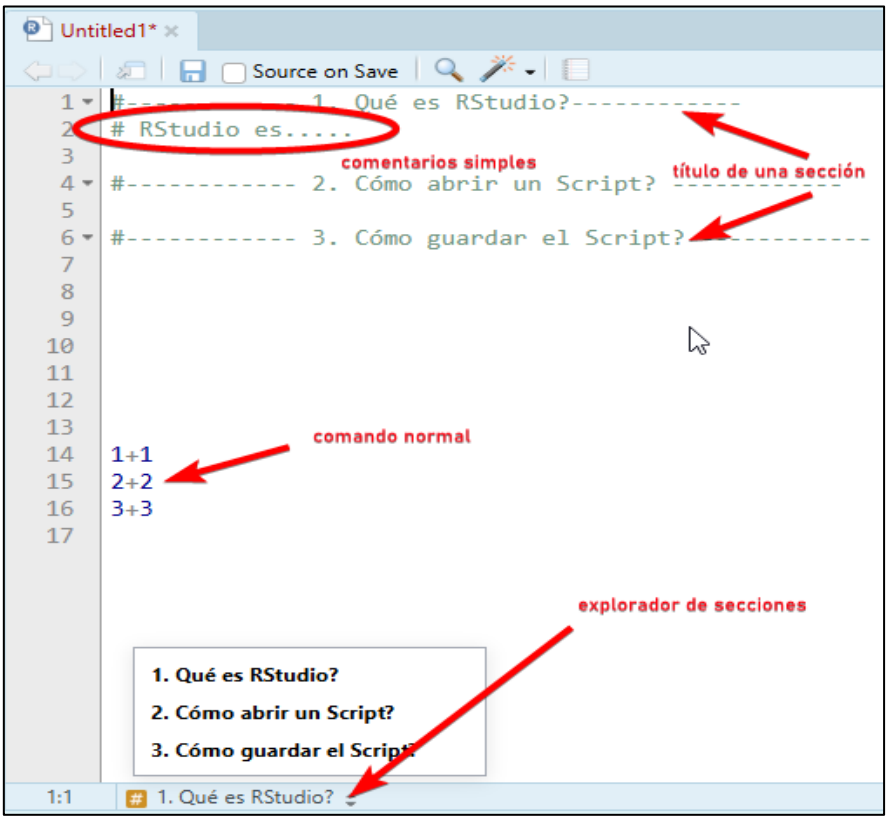
3.2.4. Comentarios y secciones

En RStudio puedes agregar comentarios dentro del código anteponiendo el símbolo # al inicio de una línea. Todo lo que siga a este carácter será ignorado por R, lo que permite incluir notas, explicaciones o recordatorios sin afectar la ejecución del programa. Esta práctica es especialmente útil en análisis extensos o cuando se trabaja en equipo, ya que ayuda a documentar y comprender mejor cada paso del script.

Además, para crear secciones navegables dentro del archivo, puedes usar un encabezado seguido por # ---- (cuatro guiones). Por ejemplo:

```
# 1. ¿Qué es RStudio? ----
# 2. ¿Cómo abrir un script? ----
# 3. ¿Cómo guardar un script? ----
```

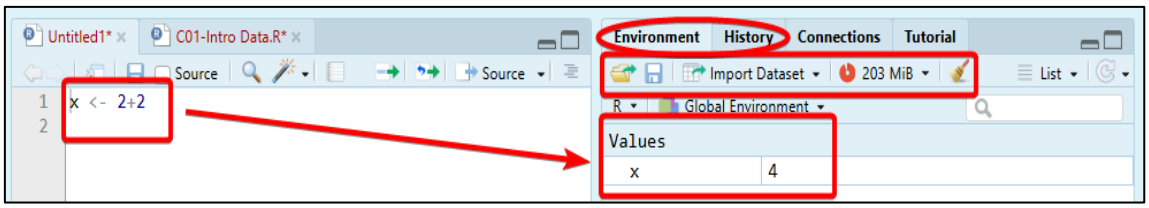
Esto permite organizar el código y moverte fácilmente entre las distintas partes del proyecto.



3.3. Entorno

El panel de entorno está compuesto por dos pestañas principales: **Environment** y **History**.

En el ***Environment*** muestra todos los objetos creados durante la sesión, como data frames, vectores o funciones. Desde su barra de opciones también puedes cargar o guardar una sesión de trabajo, importar datos y eliminar objetos del entorno para mantenerlo organizado.



En la pestaña **History** se registran las instrucciones ejecutadas. Como opciones, podemos cargar y guardar el historial de la sesión, seleccionar una o más instrucciones y enviarlas bien a la consola bien al script, y limpiar el historial.

3.4. Archivos / Gráficos / Paquetes / Ayuda

En este panel cabe destacar las siguientes pestañas, cada una con diferentes opciones:

- **Files:** funciona como un explorador de archivos, permitiendo navegar entre carpetas, abrir documentos y gestionar los ficheros del proyecto.
- **Plots:** muestra los gráficos generados durante la sesión. Ofrece opciones útiles como:
  - **Zoom:** para ampliar el gráfico en una ventana separada.
  - **Export:** para guardar la visualización como imagen, PDF o copiarla al portapapeles.
- **Packages:** presenta la lista de paquetes instalados en R y aquellos que están cargados en la sesión. Desde esta pestaña puedes instalar nuevos paquetes, activar o desactivar los existentes y actualizar versiones.
- **Help:** permite acceder rápidamente a la documentación de cualquier función o paquete, facilitando la consulta de ejemplos, argumentos y explicación detallada de su uso.

4. Tipos de Bases de Datos

Las bases de datos pueden clasificarse según su estructura temporal y la forma en cómo se organiza su información. En el análisis estadístico y econométrico, esta clasificación es fundamental, ya que determina el tipo de metodología que puede aplicarse y las conclusiones que pueden obtenerse. En este sentido, las bases de datos suelen agruparse en tres tipos principales: corte transversal, series de tiempo y datos de panel[4].

4.1. Datos de corte transversal

Una base de datos tipo corte transversal se caracteriza por tener información de un conjunto de individuos, hogares, empresas, regiones, países u otras unidades, en un punto datos en el tiempo. Es preciso mencionar que no todos los datos pueden pertenecer a un mismo periodo de tiempo, es posible que la información sea recolectada en diferentes semanas o meses.

En términos de estructura, este tipo de base se organiza de manera que cada fila representa una unidad de análisis y cada columna corresponde a una variable (edad, ingreso, nivel educativo, gasto, etc.). No existe una dimensión temporal repetida para las mismas unidades, lo que significa que cada unidad aparece una sola vez en la base.

Por ejemplo, se aplica una encuesta en el año 2024 a un grupo de agricultores ubicados en diferentes regiones del país para recopilar información sobre su nivel de producción, ingresos, costos y acceso a crédito, cada agricultor será registrado una sola vez en la base de datos. Esta información permitirá analizar diferencias productivas y económicas

[4] Wooldridge, J. M. (2010).

entre regiones en ese periodo específico, aunque no permitirá observar cambios en el tiempo para los mismos productores.

Agricultor	Región	Ingresos Mensual (S/)	Costos Mensuales (S/)	Acceso a Crédito
A	Cusco	3,500	2,100	Sí
B	Cajamarca	4,200	2,800	No
C	Junín	2,900	1,900	No
D	Ayacucho	5,000	3,200	Sí
E	Cajamarca	3,800	2,400	Sí
F	Junín	2,600	1,700	No
G	Cusco	4,500	2,900	Sí
H	Ayacucho	3,200	2,000	No
I	Cajamarca	5,400	3,600	Sí
J	Ayacucho	2,800	1,850	No

En cuanto a las técnicas econométricas de análisis, los datos de corte transversal suelen emplearse en modelos econométricos como la regresión por Mínimos Cuadrados Ordinarios (OLS), modelos logit y probit para variables dependientes binarias, modelos multinomiales y técnicas de análisis multivariado. Estos métodos permiten estudiar relaciones entre variables y estimar asociaciones o efectos en un periodo específico.

4.2. Datos de series de tiempo

Una base de datos de series de tiempo se caracteriza por contener información de una misma unidad de análisis en distintos momentos del tiempo. Esta unidad puede ser un país, una región, una empresa o cualquier variable agregada cuyo comportamiento se desea analizar a lo largo de periodos sucesivos (mensuales, trimestrales, anuales, etc.).

En términos de estructura, cada fila representa un periodo de tiempo y cada columna corresponde a una variable (por ejemplo, PBI, inflación, tipo de cambio, empleo, gasto público, entre otras). A diferencia del corte transversal, aquí sí existe una dimensión temporal explícita, ya que la misma unidad es observada repetidamente en el tiempo.

Por ejemplo, si se analiza la producción agrícola anual del Perú desde 2010 hasta 2024, cada observación corresponderá a un año distinto para la misma unidad de análisis (el país). Este tipo de datos permite estudiar tendencias, ciclos, estacionalidad y relaciones dinámicas en el tiempo.

Año	Producción Agrícola (Miles Ton.)	PBI Agrícola (Millones S/)	Inflación (%)	Crédito Agrario (Millones S/)
2014	18,500	22,300	3.2	4,500
2015	19,200	23,100	3.5	4,800

2016	20,100	24,400	2.9	5,200
2017	19,800	24,000	3.8	5,000
2018	21,000	25,600	2.1	5,700
2019	21,500	26,200	2	6,000
2020	20,700	24,800	1.8	6,300
2021	22,300	27,500	4	6,800
2022	23,000	28,900	6.5	7,200
2023	22,800	28,100	5.2	7,500
2024	24,100	30,200	3.7	8,000

En cuanto a las técnicas econométricas de análisis, los datos de series de tiempo suelen emplearse en modelos que permiten capturar la dinámica temporal de las variables, como los modelos autorregresivos (AR), de medias móviles (MA), ARIMA, modelos VAR y regresiones con variables rezagadas. Asimismo, se utilizan técnicas para analizar tendencias, estacionalidad y ciclos económicos, así como pruebas de estacionariedad y cointegración. Estos métodos permiten estudiar la evolución de las variables en el tiempo, identificar relaciones dinámicas y realizar proyecciones o pronósticos.

4.3. Datos Panel o Longitudinales

Una base de datos de panel combina las características del corte transversal y las series de tiempo, ya que contiene información de múltiples unidades de análisis observadas en distintos momentos del tiempo. Es decir, se dispone de datos para varios individuos, hogares, empresas, regiones o países, y cada uno de ellos es seguido a lo largo de varios periodos.

En términos de estructura, cada fila representa una unidad de análisis en un periodo específico, por lo que una misma unidad aparece repetidamente en la base en distintos años o momentos. En este tipo de datos existen dos dimensiones: una dimensión transversal (las unidades) y una dimensión temporal (los periodos). Las columnas corresponden a las variables de interés (ingresos, costos, producción, acceso a crédito, etc.), además de las variables identificadoras como el código de la unidad y el año.

Por ejemplo, si se analiza un grupo de agricultores ubicados en distintas regiones del país durante el periodo 2020–2024, cada agricultor será observado en varios años consecutivos. Esto permitirá estudiar no solo las diferencias entre agricultores o regiones, sino también cómo cambian sus ingresos, costos o acceso a crédito a lo largo del tiempo.

Agricultor	Año	Región	Ingresos Mensual (S/)	Costos Mensuales (S/)	Acceso a Crédito
A	2020	Cusco	3,200	2,000	Sí
A	2021	Cusco	3,400	2,050	Sí
A	2022	Cusco	3,600	2,200	Sí
B	2020	Cajamarca	4,000	2,600	No
B	2021	Cajamarca	4,300	2,700	No
B	2022	Cajamarca	4,500	2,850	Sí
C	2020	Junín	2,700	1,800	No
C	2021	Junín	2,900	1,850	No
C	2022	Junín	3,100	1,950	No

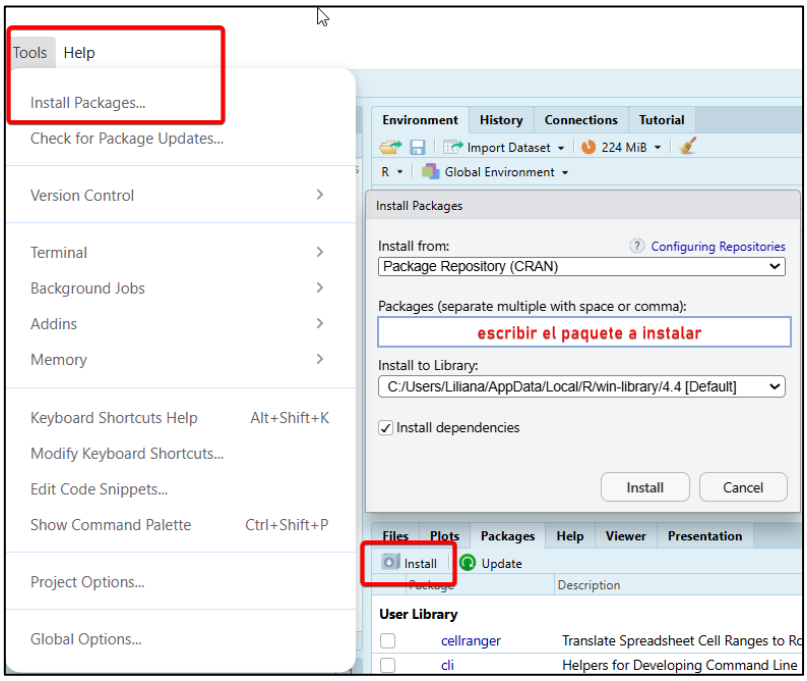
En cuanto a las técnicas econométricas de análisis, los datos de panel suelen emplearse en modelos de efectos fijos y efectos aleatorios, así como en modelos dinámicos de panel como Arellano-Bond. Asimismo, este tipo de datos es ampliamente utilizado en evaluaciones de impacto, mediante metodologías como Diferencias en Diferencias (DiD), ensayos controlados aleatorios (RCT) cuando se dispone de seguimiento en el tiempo, y modelos de variables instrumentales para corregir problemas de endogeneidad.

5. Instalar y cargar paquetes

R incluye un sistema base con funciones esenciales, pero para ampliar sus capacidades es necesario **instalar paquetes adicionales**. Estos paquetes permiten realizar tareas más avanzadas como análisis estadístico especializado, visualización de datos, modelado o conexión con bases de datos.

Podemos instalar paquetes de varias formas:

1) **Desde el cuarto panel de la interfaz**, en la pestaña *Packages*, donde puedes buscarlos, instalarlos y actualizarlos fácilmente.



2) **Utilizando la función install.packages(),** para instalar un paquete, basta con usar la función e incluir el nombre entre comillas:

```
install.packages("dplyr")
```

3) **Instalar varios paquetes a la vez,** si necesitas varios paquetes para tu sesión, puedes instalarlos en un solo paso utilizando un vector de nombres:

```
install.packages(c("dplyr", "ggplot2", "rio"))
```

Una vez instalado el paquete, hay que cargarlo para poderlo utilizar:

```
library(dplyr) # el nombre del paquete no se pone entre comillas
```

Realizar una actualización del software con **updateR**. Esta función es parte del paquete installr y funciona solamente para el sistema operativo Windows.

```
install.packages("installr")
library(installr)
updateR()
```

Actualizar de manera rápida y simultánea todos los paquetes que estén instalados en el disco duro.

```
update.packages()
```

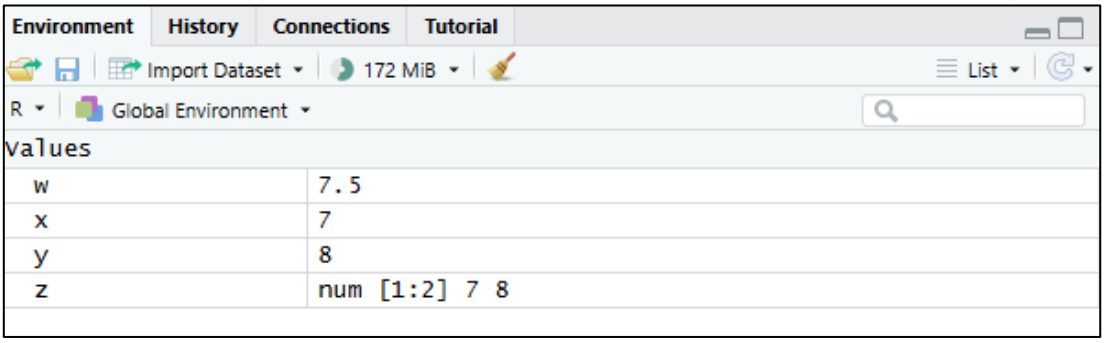
6. Introducción al lenguaje R desde RStudio

6.1. Concepto de objetos y asignación (<-).

En este primer ejercicio abordaremos los conceptos fundamentales del lenguaje R, esenciales para cualquier análisis. Veremos cómo realizar operaciones aritméticas básicas, asignar resultados a objetos usando el operador <-, combinar valores en vectores, calcular promedios y redondear resultados. También revisaremos la estructura general de las funciones en R y cómo visualizar los objetos creados en el entorno de trabajo.

Este ejercicio inicial permitirá familiarizarse con la sintaxis, los comentarios y el flujo básico de ejecución en R y RStudio, creando una base sólida para avanzar hacia análisis más complejos.

```
# Operaciones simples
3 + 4 # Resultado  7
log(10) # Resultado 2.302585
# Asignación de valores a objetos
x <- 3 + 4      # x es un objeto (vector) con valor 7
x               # muestra el contenido de x
y = 2 + 6       # también se puede usar =, aunque se recomienda <-
# Crear un vector con c()
z <- c(x, y)
# Calcular el promedio (mean)
mean(z) # Solo 1 argumento
# Guardar el resultado del promedio en otro objeto
w <- mean(z)
# Redondear el promedio
round(w, digits = 0) # 2 argumentos
# Mostrar objetos en la consola
x
print(x)
(x <- 3 + 4)
```



6.2. Tipos de objetos

En R, la información se almacena en objetos, los cuales adoptan distintas estructuras según el tipo de datos que contienen. Entre los más comunes se encuentran los vectores, matrices, listas, data frames y factores. Cada uno cumple un propósito específico: desde almacenar valores simples hasta gestionar datos tabulares o categorías. Comprender sus características y usos básicos es esencial para trabajar de manera eficiente en R.

6.2.1. Vectores:

Los vectores son la estructura de datos más básica y fundamental en R. Un vector es simplemente una colección ordenada de elementos del mismo tipo, como números, caracteres o valores lógicos.

Para crear vectores, se utiliza la función c() (de *combine*), que permite unir varios valores en un solo objeto. Por ejemplo:

- Creación de vectores con la función c():

```
# La función c() (combine) se utiliza para crear variantes
x <- c(1, 2, 3, 4)
x   ## [1]  1 2 3 4
y <- c(5, 6, 7, 8)
y   ## [1]  5 6 7 8
# Las operaciones en R son vectoriales: se realizan componente a componente.
z <- x + y
y   ## [1]  6 8 10 12
```

- Longitud de vectores:

```
# Creamos dos vectores de diferente longitud
x <- c(1, 2, 3, 4)
y <- c(1, 2, 3)
# Comprobando sus longitudes
length(x)    # devuelve 4
length(y)    # devuelve 3
# Realizamos una operación entre vectores de longitudes distintas
z <- x + y
## Warning: longitude de objeto mayor no es múltiplo de la longitud de uno menor
z      ## [1] 2 4 6 5
# R muestra un warning pero realiza la operación reciclando los elementos del vector más corto
```

- Coerción implícita y explícita:

```
# Coerción implícita: R convierte automáticamente todos los elementos al mismo tipo
x <- c(1, 2, "a")
y <- c(FALSE, 1)
z <- c("a", TRUE)
class(x)   ## [1] "character"
class(y)   ## [1] "numeric"
class(z)   ## [1] "character"
## Coerción explícita: el usuario fuerza el tipo de dato
x <- c(1, 2, "a")
as.numeric(x)   ## [1]  "1" "2" "a"
# Comprobación del tipo de objeto
is-numeric(x)
is-character(x)
```

- Acceder a elementos de un vector:

```
# Para acceder o seleccionar componentes se usan los corchetes [ ]
x <- c(2, 4, 6, 8)
# Accedemos a elementos
x[2]   ## [1] 4
x[4]   ## [1] 8
x[-2]   ## [1] 2 6 8 excluye el segundo elemento
length(x) ## [1] 4 número de elementos
```

- Listar y borrar objetos:

```
# Listar los objetos existentes en el entorno
ls()          ## [1] "x" "y" "z"
objects(x)  ## [1] "x" "y" "z"
# Borrar un objeto específico
rm(z)
ls()
# Borrar todos los objetos
rm(list = ls())
ls()          ## character(0)
```

6.2.2. Matrices:

Las matrices en R son estructuras de datos bidimensionales que almacenan información en filas y columnas. A diferencia de los data frames, una matriz solo puede contener elementos de un mismo tipo de dato (numéricos, lógicos, caracteres, etc.). Esto las convierte en una herramienta ideal para operaciones matemáticas, álgebra lineal y transformaciones que requieren uniformidad en los datos.

Para crear una matriz se utiliza la función matrix(), donde se especifican los valores a incluir y el número de filas y columnas deseadas.

- Crear y trabajar con matrices:

```
# Si queremos ayuda sobre esta función:
?matrix  # muestra la documentación de la función matrix()
# Vamos a crear una matriz de 2x2.
x <- matrix(data = c(1, 2, 3, 4), nrow = 2, ncol = 2)
x
# Observa que la matriz se rellena por columnas.
# Si queremos rellenarla por filas, usamos el argumento byrow = TRUE.
y <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE)
y <- matrix(c(1, 2, 3, 4), 2, 2, byrow = TRUE) # otra forma
y <- matrix(c(1, 2, 3, 4), 2, byrow = TRUE) # otra forma
y
# Otro ejemplo con más columnas:
x <- matrix(c(1, 2, 3, 4, 5, 6), ncol = 3)
x
```

- Nombres en filas y columnas:

```
# Podemos asignar nombres a las filas y columnas con el argumento dimnames.
y <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE,
            dimnames = list(c("X1", "X2"), c("Y1", "Y2")))
y

# También se pueden agregar o modificar nombres después de crear la matriz.
y <- matrix(1:10, nrow = 5)
colnames(y) <- c("Variable 1", "Variable 2")
rownames(y) <- c("obs1", "obs2", "obs3", "obs4", "obs5")
y
```

- Dimensión de una matriz:

```
# La función dim() devuelve las dimensiones (filas y columnas) de una matriz.
z <- matrix(1:20, 4)
dim(z)                  # devuelve 4 filas y 5 columnas
```

- Añadir filas y columnas:

```
# Las funciones rbind() y cbind() permiten añadir filas o columnas.
x <- matrix(c(1, 2, 3, 4), 2, 2)
y <- c(5, 6)
# Añadir por filas:
z <- rbind(x, y)
# Añadir por columnas:
z <- cbind(x, y)
# Si las dimensiones no son compatibles, R mostrará un aviso (warning):
x <- c(4, 5)
y <- c(10, 11, 12)
z <- rbind(x, y)
# También podemos crear matrices a partir de vectores con dim():
w <- 1:10
dim(w) <- c(2, 5)
```

- Seleccionando elementos de una matriz:

```
# Se utilizan corchetes [] indicando la posición [fila, columna].
A <- matrix(1:16, 4, 4)
A
# Ejemplos de selección:
A[2, 3]          # elemento en fila 2, columna 3
A[c(1, 2), c(2, 4)]  # filas 1-2 y columnas 2-4
A[1:3, 2:4]      # submatriz
A[1, ]           # primera fila
A[, 2:3]         # columnas 2 y 3
A[-c(1, 3), ]    # todas las filas menos la 1 y 3
```

6.2.3. Listas:

Las **listas** en R son estructuras de datos muy flexibles, ya que, a diferencia de los vectores o las matrices, permiten almacenar elementos de distintos tipos dentro de un mismo objeto. Una lista puede contener números, caracteres, valores lógicos, vectores completos, matrices, data frames e incluso otras listas. Cada elemento dentro de la lista se denomina componente y puede ser accedido, modificado o nombrado individualmente.

Las listas son especialmente recomendadas cuando se necesita agrupar información diferente en un solo lugar. Por ejemplo, si quieres guardar en un mismo objeto el nombre

de una persona, su edad y un vector con sus notas, una lista te permite combinar todos esos elementos, aunque sean de tipos distintos.

```
# Creamos una lista con 5 componentes de diferentes tipos:
x <- list(c(1,2,3,4), "Curso", FALSE, 1+2i, 3L)
x
x[[3]]       # accede al tercer componente (el valor lógico FALSE)
x[[1]][2]    # accede al segundo elemento del primer componente (2)
y <- list(
  Titulacion = c("Economía", "Sociología", "Derecho"),
  Edad = c(25, 26, 27)
)
y$Titulacion        # accede al componente "Titulacion"
y[[1]]              # también accede al primer componente
y[1]                # devuelve una sublista con ese componente
y[[1]][1]           # primer elemento dentro de "Titulacion"
y$Titulacion[1]    # equivalente
# Si el componente es numérico, podemos realizar operaciones matemáticas:
y[[2]] * 3          ## [1] 75 78 81
z <- vector("list", length = 3) # También podemos crear una lista vacía
z # Esto genera una lista con tres posiciones vacías, listas para llenarse.
```

6.2.4. Data frame:

Los data frames son una de las estructuras más utilizadas en R, ya que permiten almacenar información en formato tabular, organizada en filas (observaciones) y columnas (variables), de manera similar a hojas de cálculo como Excel o bases de datos en SPSS y Stata.

A diferencia de las matrices, los data frames pueden contener distintos tipos de datos en cada columna: valores numéricos, texto, fechas, factores o valores lógicos. Esto los convierte en una estructura ideal para trabajar con datos reales, donde cada variable puede tener un tipo diferente.

Además, los data frames permiten acceder fácilmente a columnas por nombre, filtrar filas, agregar nuevas variables y aplicar funciones por columnas, lo que los hace esenciales en la mayoría de los análisis estadísticos y proyectos de ciencia de datos en R.

- Crear un data frame de ejemplo:

```
# Crear un data frame de ejemplo:
x <- data.frame(
  Titulacion = c("Economía", "ADE", "Sociología", "Magisterio"),
  Edad = c(25, 27, 25, 24)
)
x
class(x) # Verificamos la clase del objeto ("data.frame")
# Número de filas y columnas
nrow(x)  # 4 filas
ncol(x)  # 3 columnas
dim(x)   # 4 2 : filas y columnas
```

- Acceso a elementos del Data Frame:

```
# Acceder a una variable específica
x$Titulacion
x[1]  # también muestra la primera columna
# Acceder a los dos primeros elementos de una variable
x$Titulacion[1:2] ## "Economía" "ADE"
# Usar attach() para acceder directamente a las variables
attach(x)
Titulacion
detach(x) # se recomienda usar detach() al finalizar
```

- Añadir variables a un Data Frame:

```
# Podemos añadir nuevas columnas de distintas formas:
# 1. Asignando directamente con $
x$id <- 1:4
x
# 2. Combinando objetos con cbind()
obs <- 1:4
x <- cbind(obs, x)
x
```

- Ver el contenido: head() y tail():

```
# Los data frames suelen ser grandes, por eso usamos head() y tail() para ver las primeras o últimas observaciones.
data(EuStockMarkets)  # Datos integrados de R
head(EuStockMarkets)  # Primeras 6 observaciones por defecto
tail(EuStockMarkets)  # Últimas 6 observaciones
# También podemos especificar cuántas queremos ver:
head(EuStockMarkets, 10)
tail(EuStockMarkets, 10)
```

- Nombres de filas y columnas:

```
# Si las columnas o filas no tienen nombres, podemos asignarlos con names()
lista2 <- list(
  c("Economía", "ADE", "Sociología", "Magisterio"),
  c(25, 27, 25, 24)
)
z <- data.frame(lista2)
names(z) <- c("Titulación", "Edad") # Asignamos nombres de columna
z
```

- Eliminación de valores NA:

```
# Los valores perdidos se denotan por NA.
# Podemos detectarlos y eliminarlos fácilmente.
x <- c(1, 2, NA, NA, 5)
malos <- is.na(x)
malos # [1] FALSE FALSE  TRUE  TRUE FALSE
x[!malos] # selecciona los valores que no son NA
# Ejemplo con dos vectores
x <- c(1, 2, NA, 4, NA, 6)
y <- c("a", "b", NA, "d", NA, "f")
completos <- complete.cases(x, y)
completos # [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE
x[completos] # [1] 1 2 4 6
y[completos] # [1] "a" "b" "d" "f"
```

- Selección de datos (Subsetting):

```
# Seleccionar columnas específicas: datos es un data.frame precargado
datos2 <- datos[, 1:3]
datos3 <- datos[, c(1, 2, 4)]
# Seleccionar filas específicas:
datos4 <- datos[1:6, ]
datos5 <- datos[seq(1, nrow(datos), 5), ]  # cada 5 filas
# Seleccionar filas y columnas a la vez:
datos6 <- datos[seq(1, nrow(datos), 5), c(1, 2, 4)]
# Filtrar con condiciones lógicas:
datos7 <- datos[datos$Wind <= 4, c(1, 2)]
datos8 <- datos[datos$Wind >= 2 & datos$Wind <= 5.1, c(1, 2)]
datos9 <- datos[datos$Wind == 4, c(1, 2)]
# Usar subset() como alternativa más legible:
datos10 <- subset(datos, Month == 5 & Day <= 15, select = c(Ozone, Solar.R, Temp))
datos11 <- subset(datos, Month != 5 & Day <= 15)
```

6.2.5. Factores:

Los factores son objetos en R diseñados para representar variables cualitativas o categóricas, como sexo, nivel educativo, grupos etarios, marcas o tipos de producto. Internamente, un factor almacena las categorías como valores numéricos, pero mantiene una etiqueta para cada nivel, lo que permite trabajar con categorías de forma eficiente y coherente.

Los factores pueden ser nominales (sin un orden específico, como "Rojo", "Azul", "Verde") o ordenados (con un orden natural, como "Bajo", "Medio", "Alto"). Esta distinción es importante porque muchas funciones estadísticas y modelos utilizan esta información para realizar análisis adecuados.

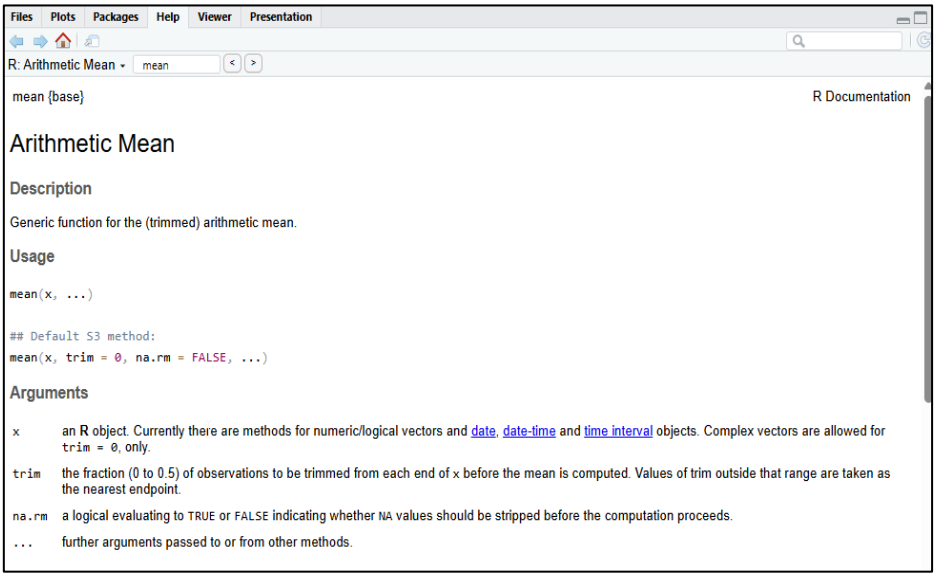
Los factores son especialmente útiles al manejar encuestas, clasificaciones o cualquier variable que represente grupos, permitiendo una gestión más clara y una correcta interpretación de resultados.

```
factor_nominal <- factor(rep(c("Ford", "Seat", "Renault"), 10))
factor_nominal
# Mostrar los niveles del factor
levels(factor_nominal)
# Por defecto, R ordena los niveles alfabéticamente
# Podemos definir el orden de los niveles manualmente usando el argumento "levels"
nuevo_factor_nominal <- factor(factor_nominal,
                               levels = c("Seat", "Renault", "Ford"))
levels(nuevo_factor_nominal)    ## [1] "Seat"    "Renault" "Ford"
# Ahora los niveles aparecen en el orden que especificamos.
```

En los modelos estadísticos, como las regresiones, es fundamental que las variables categóricas estén definidas como factores. Esto permite que R las interprete correctamente, ya que internamente asigna un valor numérico a cada nivel (1, 2, 3, …) sin perder la información cualitativa. De esta manera, los modelos pueden incorporar adecuadamente las categorías y generar estimaciones coherentes.

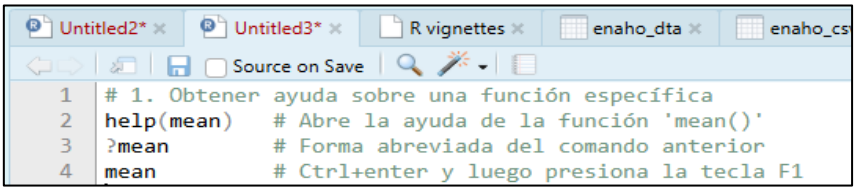
7. Soporte y documentación

El mejor amigo de un usuario de R es la ventana 'Help'.

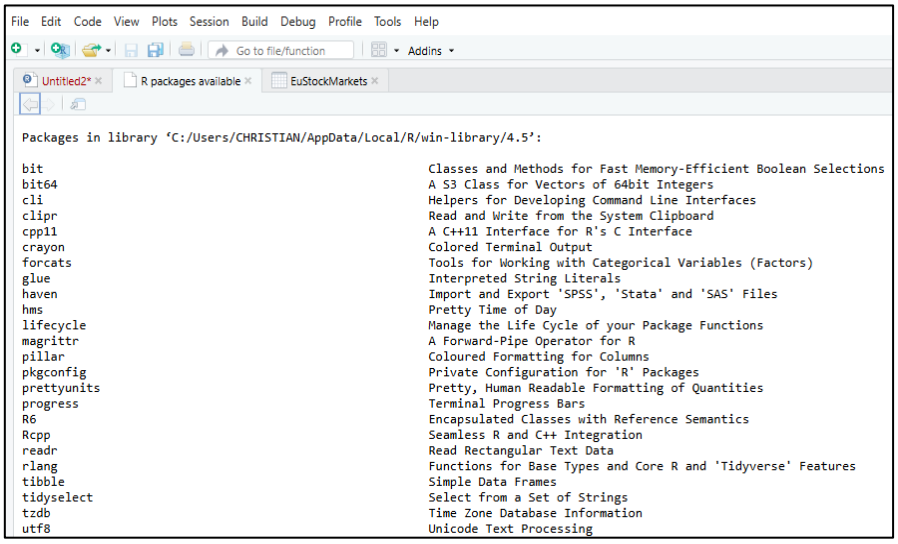


En muchas ocasiones necesitamos ayuda sobre cómo funciona una determinada función, cuáles son sus argumentos, sobre algún paquete, etc. Para ello, R ofrece varias maneras de acceder a su ayuda integrada. Por ejemplo, podemos solicitar información sobre la función mean() directamente desde la consola.

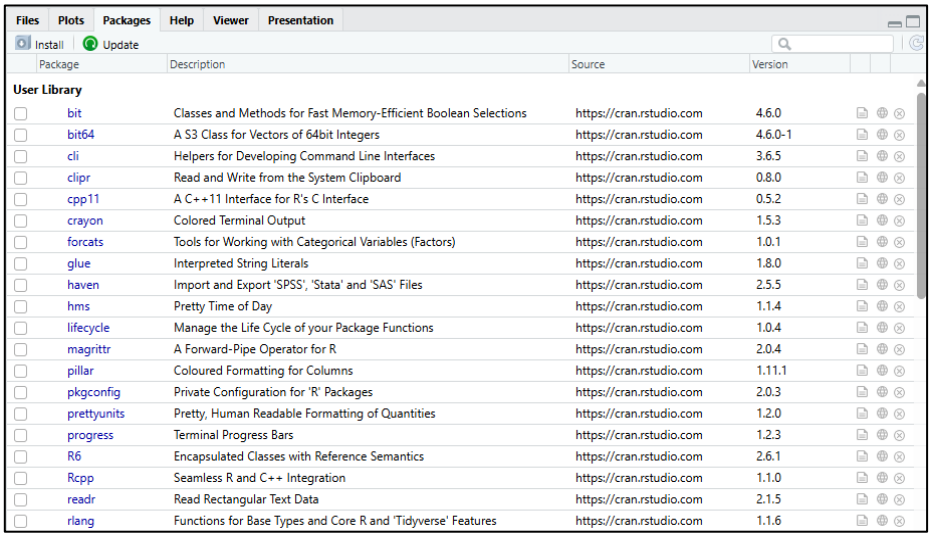
1. Obtener ayuda sobre una función específica



2. Revisar los paquetes instalados: library()



En RStudio, también puedes verlos desde la pestaña "Packages" (divididos en "User Library" y "System Library").



3. Obtener ayuda sobre un paquete específico: library(help = "foreign")

```
                Información sobre el paquete 'foreign'

Descripción:

Package:            foreign
Priority:           recommended
Version:            0.8-88
Date:               2025-01-10
Title:              Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat',
                    'Weka', 'dBase', ...
Depends:            R (>= 4.0.0)
Imports:            methods, utils, stats
Authors@R:          c( person("R Core Team", email = "R-core@R-project.org", role =
                    c("aut", "cph", "cre"), comment = c(ROR = "02zz1nj61")),
                    person("Roger", "Bivand", role = c("ctb", "cph")), person(c("Vincent",
                    "J."), "Carey", role = c("ctb", "cph")), person("Saikat", "DebRoy",
                    role = c("ctb", "cph")), person("Stephen", "Eglen", role = c("ctb",
                    "cph")), person("Rajarshi", "Guha", role = c("ctb", "cph")),
                    person("Swetlana", "Herbrandt", role = "ctb"), person("Nicholas",
                    "Lewin-Koh", role = c("ctb", "cph")), person("Mark", "Myatt", role =
                    c("ctb", "cph")), person("Michael", "Nelson", role = "ctb"),
                    person("Ben", "Pfaff", role = "ctb"), person("Brian", "Quistorff", role
                    = "ctb"), person("Frank", "Warmerdam", role = c("ctb", "cph")),
                    person("Stephen", "Weigand", role = c("ctb", "cph")), person("Free
                    Software Foundation, Inc.", role = "cph"))
Contact:            see 'MailingList'
Copyright:          see file COPYRIGHTS
Description:        Reading and writing data stored by some versions of 'Epi Info',
                    'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', and for
                    reading and writing some 'dBase' files.
ByteCompile:        yes
Biarch:             yes
License:            GPL (>= 2)
BugReports:         https://bugs.r-project.org
MailingList:        R-help@r-project.org
URL:                https://svn.r-project.org/R-packages/trunk/foreign/
NeedsCompilation:   yes
Packaged:           2025-01-10 13:45:38 UTC; hornik
```

8. Gestión de bases de datos

Esta sección aborda los procedimientos esenciales para la gestión de bases de datos en R, desde la importación y exportación de archivos hasta la manipulación, transformación y organización de variables. También incluye técnicas de filtrado, ordenamiento y unión de bases, fundamentales para preparar la información antes del análisis.

8.1. Importación de datos (.sav, .xlsx, .csv, .dta)

R permite trabajar con múltiples tipos de archivos, incluyendo datos planos (CSV, TXT), hojas de cálculo (Excel), archivos estadísticos (Stata, SPSS, SAS), así como estructuras más complejas como JSON y XML, además de conectarse a bases de datos relacionales (MySQL, PostgreSQL, SQL Server). Como ejemplo, se utilizará la base del módulo de características del hogar de la ENAHO 2024.

```
#---------- DESCARGAR DATA: .SAV ----------
install.packages("haven")   # Instala el paquete 'haven' (solo la primera vez)
library(haven)              # Carga el paquete para leer archivos SPSS, Stata y SAS
setwd("C:/Users/User/Documents/Rprueba/sav")      # Define la carpeta de trabajo
enaho_sav <- read_spss("Enaho01-2024-100.sav")  # Importa el archivo .sav (SPSS)
View(enaho_sav)                                  # Abre los datos en una pestaña tipo Excel
#---------- DESCARGAR DATA: .XLSX ----------
install.packages("readxl")    # Instala el paquete 'readxl' (solo una vez)
library(readxl)               # Carga el paquete para leer archivos Excel (.xlsx)
setwd("C:/Users/User/Documents/Rprueba/xlsx")       # Cambia el directorio de trabajo
enaho_excel <- read_excel("Enaho01-2024-100.xlsx", sheet = "DATOS", skip = 0)  # Lee la hoja "DATOS" e indica la fila encabezado
View(enaho_excel)                                # Muestra los datos en la interfaz
#---------- DESCARGAR DATA: .CSV ----------
install.packages("readr")     # Instala el paquete 'readr' (solo la primera vez)
library(readr)                # Carga el paquete para leer archivos de texto (.csv)
setwd("C:/Users/User/Documents/Rprueba/csv")      # Define el directorio donde está el CSV
enaho_csv <- read_csv("Enaho01-2024-100.csv")    # Importa el archivo CSV
View(enaho_csv)                                  # Visualiza el dataset en RStudio
#---------- DESCARGAR DATA: .DTA ----------
library(haven)                # Carga 'haven' para leer archivos .dta (Stata)
setwd("C:/Users/User/Documents/Rprueba/dta")     # Define la carpeta donde está el .dta
enaho_dta <- read_dta("Enaho01-2024-100.dta")    # Importa el archivo Stata
View(enaho_dta)                                  # Visualiza el dataset en RStudio
```

8.2. Exportación de resultados (.sav, .xlsx, .csv, .dta)

```
#----------  Exportar a SPSS (.sav)----------
write_sav(enaho_dta, "C:/Users/User/Documents/Rprueba/dta/enaho_sav")
#----------  Exportar a Excel (.xlsx)----------
install.packages("writexl")
library(writexl)
write_xlsx(enaho_dta, "C:/Users/User/Documents/Rprueba/dta/enaho_excel.xlsx")
#----------  Exportar a CSV----------
write_csv(enaho_dta, "C:/Users/User/Documents/Rprueba/dta/enaho_csv")
#----------  Exportar a STATA (.dta)----------
write_dta(enaho_dta, "C:/Users/User/Documents/Rprueba/dta/enaho_dta")
```

8.3. Creación, renombrado y etiquetado de variables

Crear, renombrar y etiquetar variables es fundamental para organizar y documentar adecuadamente una base de datos antes del análisis. Estas tareas permiten generar información derivada, mejorar la claridad de los nombres de las variables y añadir descripciones que faciliten la interpretación, especialmente cuando se trabaja con grandes conjuntos de datos o cuando los resultados serán compartidos con otros usuarios o instituciones.

Para crear variables	mutate() — *dplyr*
Para renombrar variables	rename() — dplyr
Para etiquetar variables	labelled() — haven (asignar etiquetas a valores)

```
install.packages("tidyverse")  # Solo la primera vez
library(tidyverse)
# Creamos una base simple de ejemplo
data <- tibble(
  id = 1:5,
  edad = c(22, 35, 29, 40, 31),
  ingreso = c(1500, 2500, 1800, 3000, 2300),
  genero = c("M", "F", "F", "M", "M")
)
# Crear una nueva variable
data <- data %>%
  mutate(ingreso_miles = ingreso / 1000)  # convierte a miles
# Renombrar variables
data <- data %>%
  rename(salario = ingreso)
# Etiquetar variables (usando labelled del paquete haven)
library(haven)
data$genero <- labelled(data$genero, c(Masculino = "M", Femenino = "F"))
```

8.4. Recodificación y transformación de variables

La recodificación y transformación de variables es esencial para adaptar los datos a las necesidades del análisis y garantizar su correcta interpretación. Estas operaciones permiten agrupar categorías, modificar escalas, generar versiones estandarizadas o normalizadas de las variables y ajustar valores según criterios analíticos específicos. En R, estas tareas se realizan de manera eficiente utilizando funciones del tidyverse, que facilitan la modificación precisa y reproducible de los datos.

```
# Recodificar variable "genero"
data <- data %>%
  mutate(
    genero_rec = ifelse(genero == "M", "Hombre", "Mujer"),
    grupo_edad = case_when(
      edad < 30 ~ "Joven",
      edad >= 30 & edad < 40 ~ "Adulto",
      TRUE ~ "Mayor"
    )
  )
# Transformar variable numérica (logaritmo, cuadrado, etc.)
data <- data %>%
  mutate(log_ingreso = log(salario))
```

8.5. Filtrado, ordenamiento y eliminación de variables

El filtrado, ordenamiento y eliminación de variables permite depurar y organizar los datos para enfocarse únicamente en la información relevante para el análisis. En muchas ocasiones, las bases de datos contienen información complementaria que no es necesaria para el análisis de datos. Estas operaciones ayudan a seleccionar subconjuntos específicos de observaciones, ordenar registros según criterios definidos y descartar variables innecesarias o redundantes.

```
# Filtrar solo hombres mayores de 30
data_filtrada <- data %>%
  filter(genero_rec == "Hombre", edad > 30)
# Ordenar por edad descendente
data_ordenada <- data %>%
  arrange(desc(edad))
# Seleccionar (mantener) o eliminar variables
data_reducida <- data %>%
  select(id, edad, salario)       # Mantiene solo estas
data_sin_log <- data %>%
  select(-log_ingreso)            # Elimina esa variable
```

8.6. Unión de bases de datos

La unión de bases de datos es una de las tareas más importantes en el procesamiento y análisis de información, especialmente cuando los datos provienen de diferentes fuentes o contienen variables complementarias. Unir bases permite consolidar

información, ampliar el número de variables disponibles, integrar registros adicionales y construir bases más completas para el análisis estadístico. En R, estas operaciones se realizan mediante las funciones del paquete dplyr, que ofrecen distintos tipos de uniones dependiendo de la estructura y del objetivo analítico.

- **Función merge**

La función merge() en R se utiliza para unir dos dataframes a partir de una o más variables en común (claves), de forma similar a los *joins* en SQL. Permite combinar información de distintas bases que comparten un identificador (por ejemplo, id), agregando columnas según la coincidencia de valores. Se usa cuando tienes datasets relacionados (por ejemplo, datos de pescadores y su producción) y necesitas integrarlos horizontalmente. Además, merge() ofrece flexibilidad mediante argumentos como by, all, all.x y all.y, que permiten definir el tipo de unión (inner, left, right o full join).

A continuación, se describen los principales tipos de uniones disponibles en R:

a) *left_join():* Conserva todos los registros de la base principal (a la izquierda) y agrega únicamente la información coincidente de la segunda base, según la clave definida. Se usa cuando quieres mantener toda tu base original, sin perder observaciones.

b) *inner_join():* Devuelve únicamente las observaciones que aparecen en ambas bases. Es útil cuando necesitas trabajar solo con los registros que tienen coincidencia en ambas fuentes.

c) *right_join():* Conserva todos los registros de la segunda base (a la derecha), complementando con la información coincidente de la base principal. Se emplea cuando la base secundaria debe ser priorizada o es la de referencia.

d) *full_join():* Combina todos los registros de ambas bases, independientemente de si existe o no coincidencia en la clave. Es ideal para integrar información dispersa sin descartar ningún dato.

e) *bind_rows():* Une filas de dos o más bases que comparten la misma estructura de variables. A diferencia de los joins anteriores (que combinan columnas), bind_rows() apila datos uno debajo de otro, aumentando el número de registros.

```
# Creamos una segunda base de ejemplo
data2 <- tibble(
  id = c(3, 4, 5, 6),
  ciudad = c("Lima", "Cusco", "Piura", "Trujillo")
)
# Uniones según tipo:
left_joined  <- left_join(data, data2, by = "id")          # Mantiene todos los de data
inner_joined <- inner_join(data, data2, by = "id")    # Solo los comunes
right_joined <- right_join(data, data2, by = "id")       # Mantiene los de data2
full_joined  <- full_join(data, data2, by = "id")          # Une todo
combined     <- bind_rows(data, data)                  # Une filas (igual estructura)
```

- **Función rbind**

La función rbind() en R se utiliza para unir dataframes apilándolos verticalmente, es decir, agregando nuevas filas. Para que funcione correctamente, ambas bases deben tener las mismas columnas (mismos nombres y tipos de variables). Se usa cuando tienes información separada en distintos bloques, pero con la misma estructura (por ejemplo, registros de pescadores de diferentes días o regiones) y quieres consolidarlos en una sola base. A diferencia de merge(), rbind() no busca coincidencias entre variables, simplemente concatena observaciones.

```
# Ejemplo de datos anchos (wide)
wide <- tibble(
  id = 1:3,
  ingreso_2023 = c(2000, 2500, 3000),
  ingreso_2024 = c(2200, 2700, 3100)
)
# Pasar de ancho a largo (pivot_longer)
long <- wide %>%
  pivot_longer(cols = starts_with("ingreso"),
               names_to = "año",
               values_to = "ingreso")
# Volver a formato ancho (pivot_wider)
wide2 <- long %>%
  pivot_wider(names_from = año, values_from = ingreso)
```

8.7. Medidas de tendencia central, dispersión y forma

En el análisis de datos es clave poder analizar la información mediante los estadísticos clásicos de tendencia central, dispersión y forma.

Medidas de tendencia central

- **Media:** Promedio de todos los valores; indica el valor central promedio.
- **Mediana:** Valor que divide la distribución en dos partes iguales; útil cuando hay valores extremos.
- **Moda:** Valor que aparece con mayor frecuencia en la distribución.

Medidas de dispersión

- **Desviación estándar:** Indica cuánto se alejan en promedio los valores respecto a la media.
- **Varianza:** Medida del grado de dispersión; es la desviación estándar elevada al cuadrado.
- **Rango:** Diferencia entre el valor máximo y mínimo; muestra la amplitud total de los datos.
- **IQR (Rango intercuartílico):** Distancia entre el cuartil 1 y cuartil 3; refleja la dispersión del 50% central de los datos.

Medidas de forma

- **Asimetría (Skewness):** Indica si la distribución se inclina hacia valores altos o bajos.
- **Curtosis:** Evalúa el grado de concentración o dispersión de los valores respecto al centro de la distribución.

```
# Cargar tidyverse
library(tidyverse)
# Base de ejemplo
data <- tibble(
  ingreso = c(1200, 1500, 1800, 2200, 2500, 2700, 3000)
)
# Medidas de tendencia central
mean(data$ingreso)          # media
median(data$ingreso)        # mediana
mode_val <- names(sort(table(data$ingreso), decreasing = TRUE))[1]
mode_val                    # moda
# Medidas de dispersión
sd(data$ingreso)            # desviación estándar
var(data$ingreso)           # varianza
range(data$ingreso)         # rango (mínimo y máximo)
IQR(data$ingreso)           # rango intercuartílico
# Medidas de forma
install.packages("moments")  # solo la primera vez
library(moments)
skewness(data$ingreso)      # asimetría
kurtosis(data$ingreso)      # curtosis
```

8.8. Resumen estadístico con summary() y skimr.

Ambos paquetes generan resúmenes estadísticos de forma rápida y estructurada. La función **summary()** ofrece una visión general de variables numéricas y categóricas, mostrando medidas básicas como mínimos, máximos, medianas y frecuencias. Por su parte, **skimr** amplía este resumen con estadísticas más completas y un formato más ordenado, facilitando la exploración inicial de los datos y la identificación de patrones, valores atípicos o posibles problemas en la base.

```
summary(data)  # resumen base de R
install.packages("skimr")  # solo la primera vez
library(skimr)
skim(data)     # resumen completo: n, NA, media, sd, p25, p50, p75, etc.
```

8.9. Gráficos univariantes y bivariantes con ggplot2

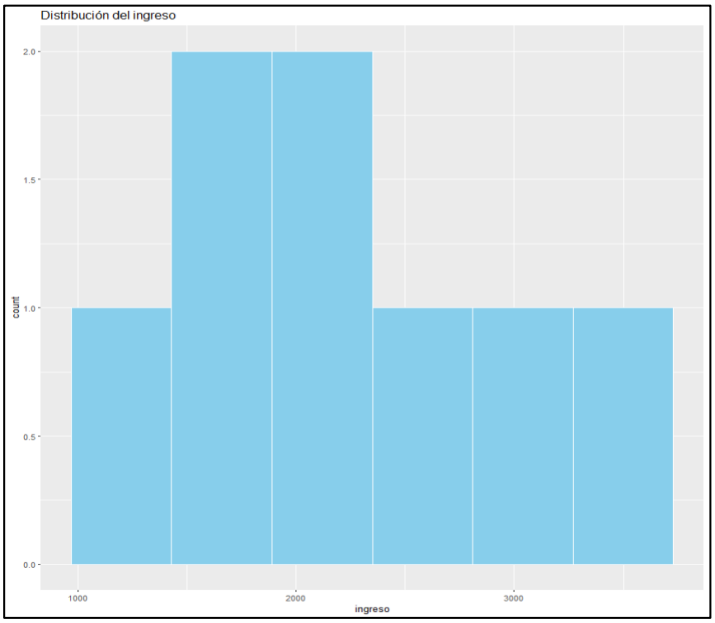
R destaca por la amplia variedad de opciones gráficas que ofrece y por la alta calidad visual de sus representaciones. En esta sección se utilizan las herramientas de ggplot2, uno de los paquetes más potentes del tidyverse, para construir gráficos univariantes y

bivariantes de manera flexible y personalizable. ggplot2 permite crear visualizaciones profesionales, adaptables a diferentes tipos de datos y necesidades analíticas, lo que facilita la exploración, interpretación y comunicación de resultados de forma clara y efectiva.

```
# Base de ejemplo más amplia
df <- tibble(
  edad = c(22, 25, 30, 35, 40, 45, 50, 55),
  ingreso = c(1200, 1500, 1800, 2000, 2300, 2700, 3000, 3500),
```

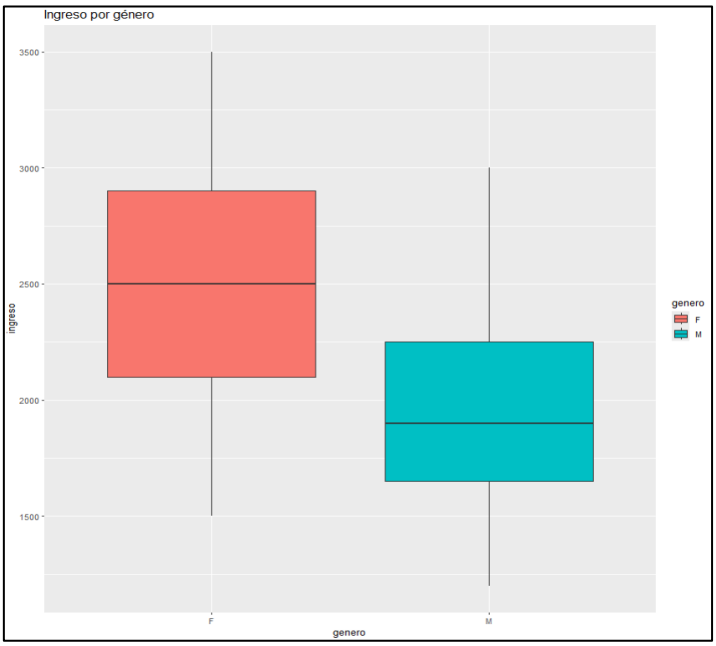
8.9.1. Histogramas

```
# Histograma (una variable numérica)
ggplot(df, aes(x = ingreso)) +
  geom_histogram(bins = 6, fill = "skyblue"  , color = "white") +
  labs(title = "Distribución del ingreso")
```



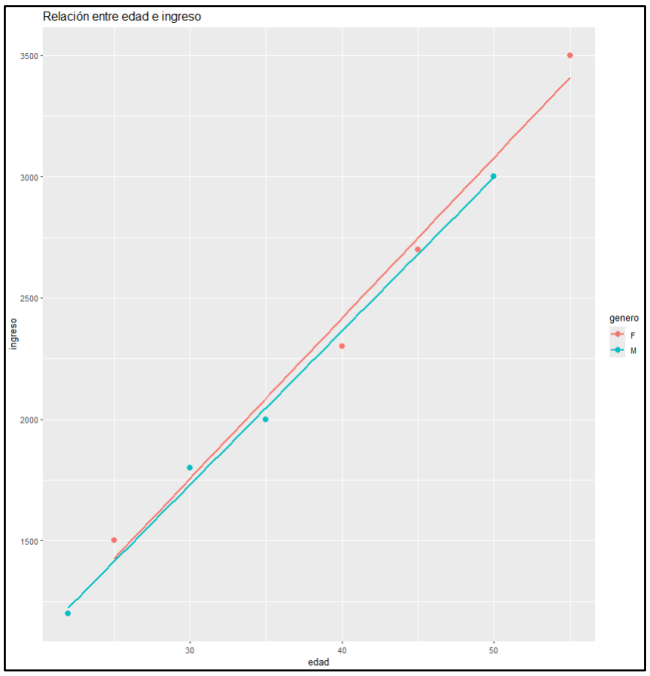
8.9.2. Boxplot por grupo

```
# Boxplot por grupo (comparativo)
ggplot(df, aes(x = genero, y = ingreso, fill = genero)) +
  geom_boxplot() +
  labs(title = "Ingreso por género")
```



8.9.3. Dispersión

```
# Dispersión (relación entre dos variables numéricas)
ggplot(df, aes(x = edad, y = ingreso, color = genero)) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Relación entre edad e ingreso")
```

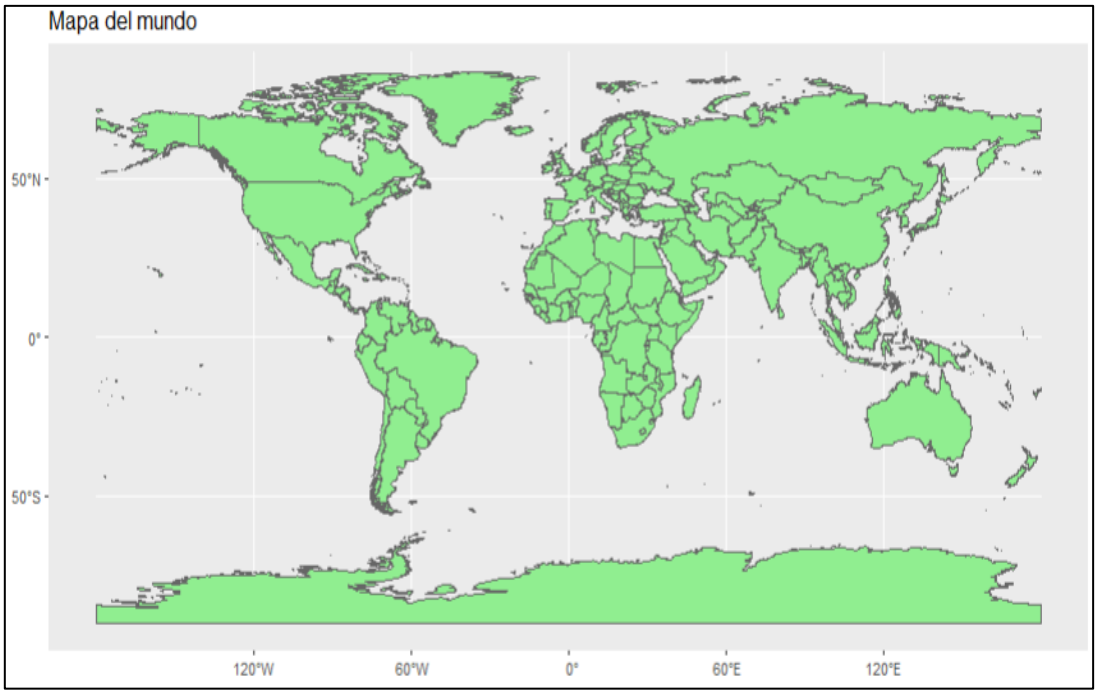


8.10. Elaboración de Mapas en R

```
# Instalamos y cargamos los paquetes necesarios
install.packages("sf")
install.packages("rnaturalearth")
install.packages("rnaturalearthdata")
library(sf)
library(rnaturalearth)
library(rnaturalearthdata)
library(ggplot2)
# Descargar un mapa del mundo
world <- ne_countries(scale = "medium", returnclass = "sf")
```

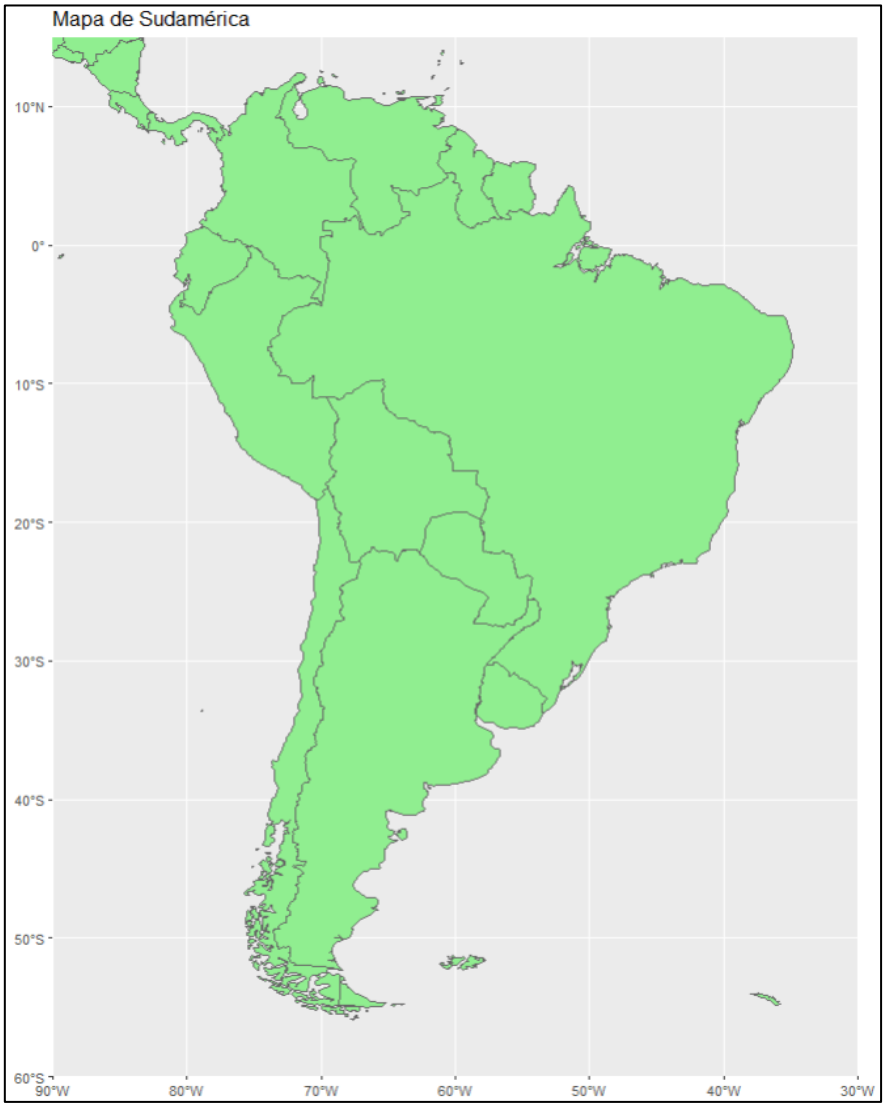
8.10.1. Mapa del mundo

```
# Mapa base simple
ggplot(data = world) +
  geom_sf(fill = "lightgreen", color = "gray40") +
  labs(title = "Mapa del mundo")
```



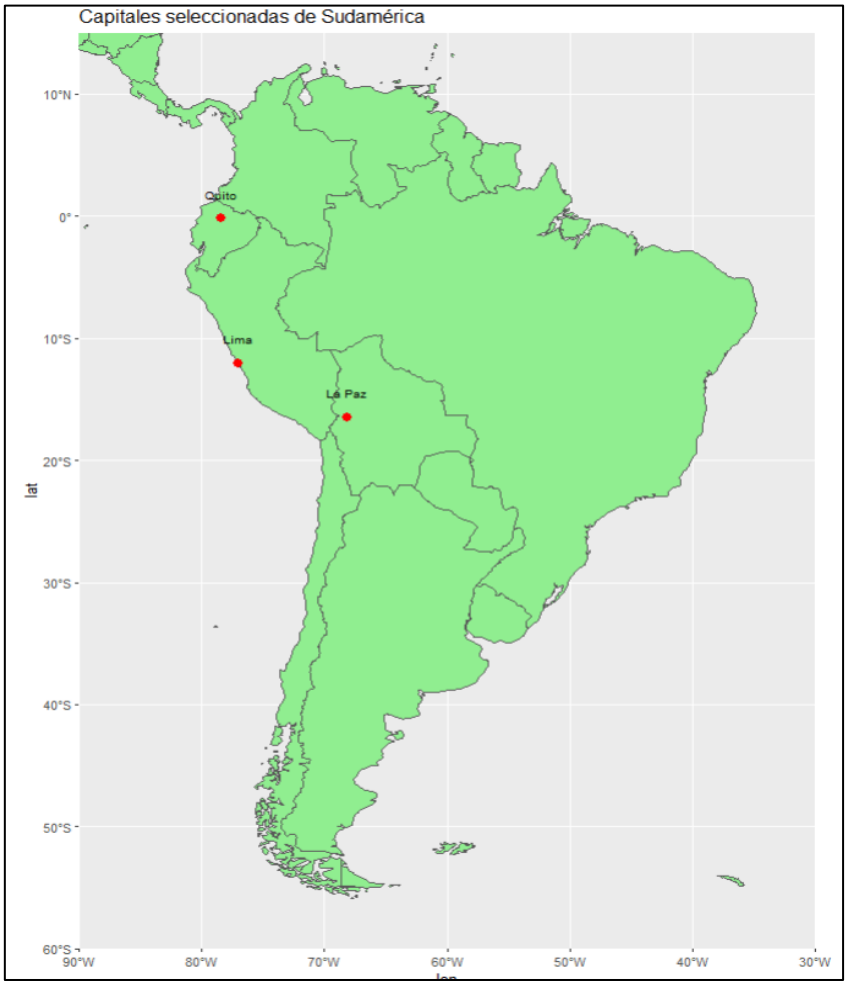
8.10.2. Mapa Sudamérica

```
# Mapa enfocado en Sudamérica
ggplot(data = world %>% filter(region_un == "Americas")) +
  geom_sf(fill = "lightgreen", color = "gray40") +
  coord_sf(xlim = c(-90, -30), ylim = c(-60, 15), expand = FALSE) +
  labs(title = "Mapa de Sudamérica")
```



8.10.3. Puntos específicos en mapa

```
# Ejemplo: agregar puntos (capitales o coordenadas)
capitales <- tibble(
  ciudad = c("Lima", "Quito", "La Paz"),
  lon = c(-77.0428, -78.4678, -68.1193),
  lat = c(-12.0464, -0.1807, -16.5000)
)
ggplot(data = world %>% filter(region_un == "Americas")) +
  geom_sf(fill = "lightgreen") +
  coord_sf(xlim = c(-90, -30), ylim = c(-60, 15), expand = FALSE) +
  geom_point(data = capitales, aes(x = lon, y = lat), color = "red", size = 3) +
  geom_text(data = capitales, aes(x = lon, y = lat, label = ciudad),
            nudge_y = 2, size = 3) +
  labs(title = "Capitales seleccionadas de Sudamérica")
```



9. Fundamentos estadísticos

9.1. Factor de expansión (survey, srvyr)

El uso del **factor de expansión** es fundamental cuando se trabaja con bases de datos provenientes de encuestas probabilísticas, ya que permite ajustar los resultados para que sean representativos de la población total. Los paquetes **survey** y **srvyr** facilitan este proceso mediante la creación de un *diseño muestral* que incorpora los pesos o factores muestrales. Con **survey**, se especifica el diseño mediante svydesign() y luego se obtienen estimaciones ponderadas como promedios o proporciones usando funciones como svymean(). Por su parte, **srvyr** ofrece una sintaxis más moderna y compatible con *dplyr*, permitiendo trabajar con tuberías y funciones como survey_mean(). En conjunto, estos paquetes permiten calcular estadísticas que reflejan adecuadamente la estructura de la encuesta y producen estimaciones válidas a nivel poblacional.

```
# Instalación y carga de paquetes
install.packages("survey")
install.packages("srvyr")
library(survey)
library(srvyr)
# Base de ejemplo
set.seed(123)
data <- data.frame(
  id = 1:100,
  ingreso = rnorm(100, mean = 2000, sd = 500),
  edad = sample(18:65, 100, replace = TRUE),
  sexo = sample(c("Hombre", "Mujer"), 100, replace = TRUE),
  factor_expansion = runif(100, 0.5, 2)  # peso o factor muestral
)
# Crear diseño muestral
design <- svydesign(ids = ~1, data = data, weights = ~factor_expansion)
svymean(~ingreso, design)
svymean(~sexo, design)
# Usando srvyr (sintaxis más moderna tipo dplyr)
design_srvyr <- as_survey_design(data, weights = factor_expansion)
design_srvyr %>%
  summarise(media_ingreso = survey_mean(ingreso),
            media_edad = survey_mean(edad))
```

9.2. Estimación puntual

```
# Base simple de ejemplo
x <- c(21, 23, 25, 22, 27, 30, 24, 26, 29, 28)
# Estimación puntual (media y varianza)
mean(x) ## [1] 25.5
var(x)  ## [1] 9.166667
```

9.3. Estimación por intervalos

```
# Estimación por intervalos (IC al 95%)
t.test(x, conf.level = 0.95)
# Ejemplo con comparación entre dos grupos
grupo <- data.frame(
  ingreso = c(rnorm(30, 2000, 300), rnorm(30, 2300, 350)),
  grupo = rep(c("Tratamiento", "Control"), each = 30))
t.test(ingreso ~ grupo, data = grupo, conf.level = 0.95)
```

9.4. Pruebas estadísticas paramétricas

Una prueba paramétrica es un tipo de prueba estadística que asume que los datos siguen una distribución específica, comúnmente una distribución normal. Además, suelen asumir:

- Homogeneidad de varianzas (grupos con varianzas similares).
- Escala de medición continua (intervalo o razón).
- Independencia entre observaciones.

```
# t de Student (2 grupos)
t.test(ingreso ~ grupo, data = grupo)
# ANOVA (más de 2 grupos)
grupo3 <- data.frame(
  ingreso = c(rnorm(20, 1800, 250),
              rnorm(20, 2000, 300),
              rnorm(20, 2300, 350)),
  sector = rep(c("A", "B", "C"), each = 20))
anova_result <- aov(ingreso ~ sector, data = grupo3)
summary(anova_result)
```

9.5. Pruebas estadísticas no paramétricas

Una prueba no paramétrica es un tipo de prueba estadística que no requiere que los datos sigan una distribución normal ni otras suposiciones fuertes. Se utiliza cuando:

- Los datos están sesgados.
- La muestra es pequeña.
- Hay outliers.
- La variable es ordinal o no se puede asumir normalidad.

```
# Mann-Whitney (equivalente no paramétrico al t-test)
wilcox.test(ingreso ~ grupo, data = grupo)
# Kruskal-Wallis (equivalente no paramétrico al ANOVA)
kruskal.test(ingreso ~ sector, data = grupo3)
```

10. Modelos de regresión y diagnóstico

Según el enfoque desarrollado por Wooldridge (2010), el modelo de regresión lineal es una herramienta econométrica utilizada para estudiar la relación entre una variable dependiente y una o más variables explicativas. Su principal objetivo es estimar el efecto parcial que tiene cada variable independiente sobre la variable de interés, manteniendo constantes los demás factores relevantes.

Este enfoque permite no solo identificar asociaciones estadísticas, sino también aproximarse al análisis causal cuando se cumplen determinados supuestos teóricos y metodológicos. Por ello, la regresión lineal es ampliamente utilizada en economía, políticas públicas y ciencias sociales para evaluar fenómenos como ingresos, empleo, consumo o productividad.

Wooldridge (2010) también enfatiza que la validez de los resultados depende del cumplimiento de ciertos supuestos del modelo, especialmente la exogeneidad de las variables explicativas, es decir, que los factores no observados no estén correlacionados con las variables incluidas en la regresión. Asimismo, destaca la importancia de revisar problemas como heterocedasticidad, multicolinealidad, errores de especificación y variables omitidas, ya que estos pueden sesgar o volver ineficientes las estimaciones. En ese sentido, el análisis econométrico no se limita a obtener coeficientes, sino que requiere interpretar resultados con criterio técnico y aplicar pruebas de diagnóstico que garanticen conclusiones confiables.

Forma matemática:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ \ldots + \beta_k x_k + u$$

Donde:

β_0: la constante o intercepto.

β_j $(j = 1, 2, \ldots, k)$: la pendiente. Es el efecto parcial sobre y cuando cambia x_j, ceteris paribus.

u: término error.

Forma matricial:

$$Y = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}, \quad X = \begin{pmatrix} 1 & X_{11} & \cdots & X_{k1} \\ 1 & X_{12} & \cdots & X_{k2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_{1n} & \cdots & X_{kn} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{pmatrix}, \quad u = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

Donde:

n: Cantidad de observaciones

k: Cantidad de variables

y	x
Dependiente	Independiente
Endógena	Exógena
Explicada	Explicativa
Respuesta	Control

10.1. Estimación por Mínimos Cuadrados Ordinarios (MCO)

El Modelo de Mínimos Cuadrados Ordinarios (MCO) es un método estadístico utilizado para estimar la relación entre una variable dependiente (Y) y una o más variables independientes (X). Su objetivo es encontrar la recta que mejor se ajusta a los datos, minimizando la suma de los cuadrados de los errores (las diferencias entre los valores observados y los predichos).

¿Qué nos permite el MCO?

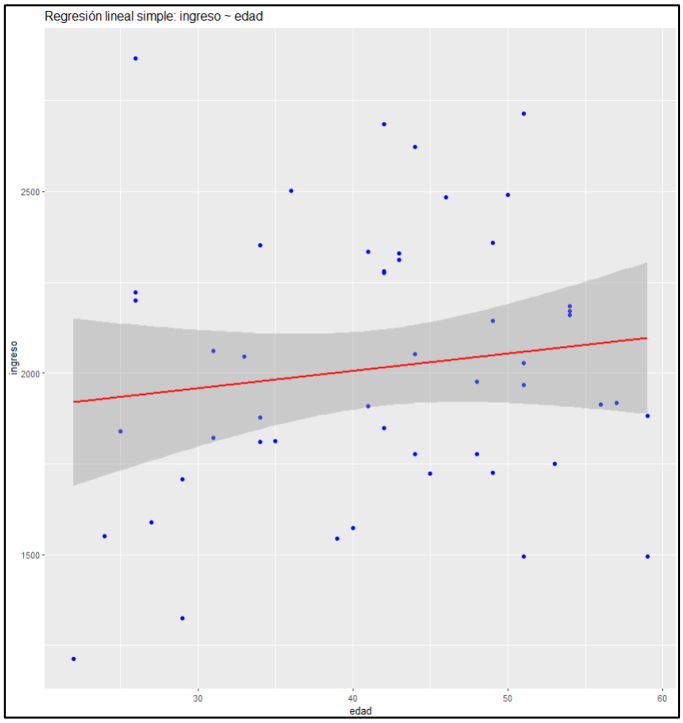
- Medir cuánto cambia Y cuando X cambia en una unidad.
- Identificar relaciones lineales entre variables.
- Hacer predicciones.
- Evaluar significancia estadística (pruebas t, F).
- Explicar variabilidad en la variable dependiente (R^2).

Supuestos clásicos del modelo MCO

Para que los estimadores sean insesgados, eficientes y consistentes (propiedades de Gauss-Markov), se deben cumplir:

1. Linealidad en los parámetros	El modelo debe ser lineal en los coeficientes (β). Las variables pueden transformarse (log, cuadrado), pero los parámetros deben entrar linealmente.
2. Muestra aleatoria	Las observaciones deben provenir de un proceso de muestreo aleatorio para garantizar representatividad y evitar sesgos sistemáticos.
3. No colinealidad perfecta entre las X	Ninguna variable explicativa puede ser combinación exacta de otra. La multicolinealidad perfecta hace imposible estimar coeficientes.
4. Esperanza del error igual a cero	El error debe tener media cero condicional a X. Garantiza que el modelo no omita variables relevantes ni esté mal especificado.
5. Homoscedasticidad	La varianza del error debe ser constante para todos los valores de X. Si no, hay heteroscedasticidad y los estimadores pierden eficiencia.
6. No autocorrelación de errores	Los errores no deben estar correlacionados entre sí. Violaciones típicas en series de tiempo (efectos persistentes). Afecta eficiencia e inferencia.
7. Normalidad del error	Los errores deben seguir una distribución normal para obtener inferencia exacta (intervalos, p-valores). No es necesaria para la insesgadez ni consistencia.

```
# Base de ejemplo
set.seed(123)
data <- data.frame(
  ingreso = rnorm(50, mean = 2000, sd = 400),
  edad = sample(20:60, 50, replace = TRUE)
)
# Estimación por Mínimos Cuadrados Ordinarios (MCO)
modelo_simple <- lm(ingreso ~ edad, data = data)
# Resumen del modelo
summary(modelo_simple)
# Visualización del ajuste
library(ggplot2)
ggplot(data, aes(x = edad, y = ingreso)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", se = TRUE, color = "red") +
  labs(title = "Regresión lineal simple: ingreso ~ edad")
```



10.2. Interpretación de coeficientes y bondad de ajuste (R^2, p-valores)

```
Call:
lm(formula = ingreso ~ edad, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-706.54 -248.06  -80.25  278.22  928.59

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1814.850    221.588   8.190 1.14e-10 ***
edad           4.775      5.168   0.924     0.36
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 370.9 on 48 degrees of freedom
Multiple R-squared:  0.01747,   Adjusted R-squared:  -0.002996
F-statistic: 0.8536 on 1 and 48 DF,  p-value: 0.3601
```

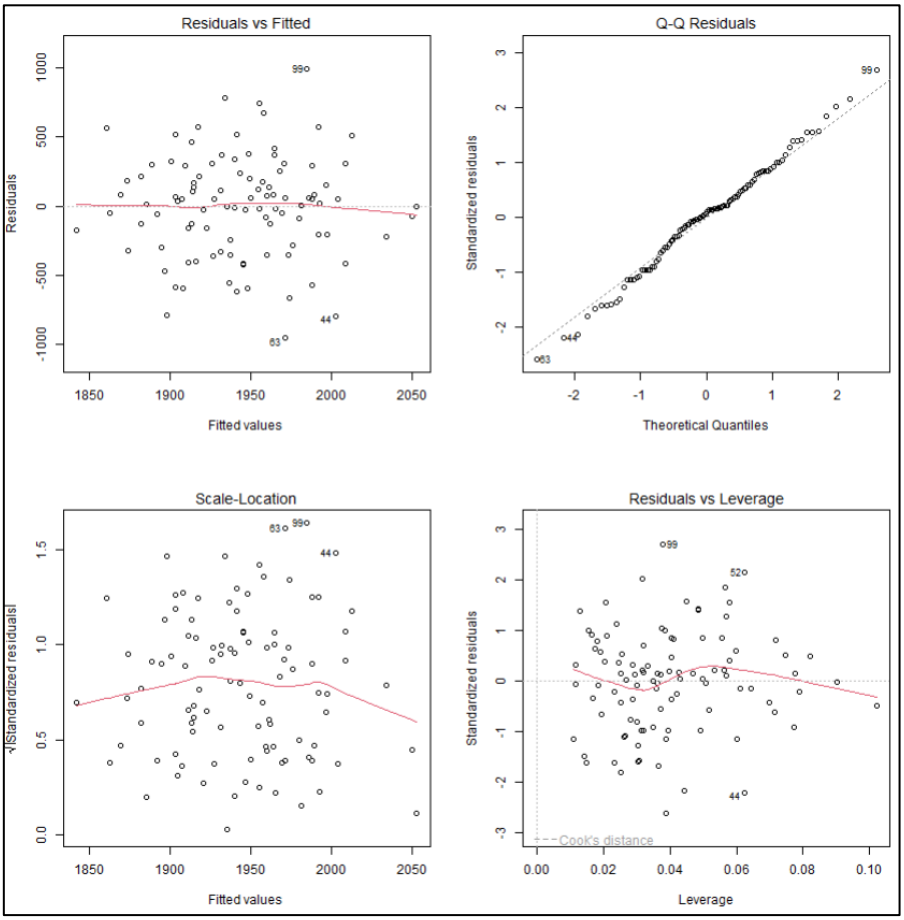
- Coeficiente de edad (4.78): Indica que, en promedio, por cada año adicional de edad, el ingreso aumenta en 4.78 unidades monetarias. Sin embargo, el valor p = 0.36 muestra que no es estadísticamente significativo, es decir, no hay evidencia suficiente para afirmar que la edad afecta el ingreso en esta muestra.
- Bondad de ajuste:

- R^2 = 0.017 → solo el 1.7% de la variación del ingreso se explica por la edad.
- R^2 ajustado ≈ 0 → el modelo prácticamente no mejora respecto a usar solo el promedio.
- F-statistic p-value = 0.36 → el modelo en conjunto no es significativo.

10.3. Evaluación del modelo y diagnóstico de errores

La evaluación del modelo de regresión múltiple permite determinar si las variables explicativas influyen significativamente sobre el ingreso y si el ajuste es adecuado. A través del *summary()* se analizan los coeficientes, su significancia estadística y la bondad de ajuste mediante el R^2 ajustado. Posteriormente, el diagnóstico gráfico de residuos permite verificar el cumplimiento de los supuestos del MCO, como la homocedasticidad, normalidad y ausencia de patrones sistemáticos. Estos pasos son fundamentales para asegurar la validez e interpretación correcta del modelo.

```
# Base extendida
data2 <- data.frame(
  ingreso = rnorm(100, 2000, 400),
  edad = sample(20:60, 100, replace = TRUE),
  educacion = sample(6:18, 100, replace = TRUE),
  horas_trab = rnorm(100, 40, 5)
)
# Modelo de regresión múltiple
modelo_multiple <- lm(ingreso ~ edad + educacion + horas_trab, data = data2)
summary(modelo_multiple)
# Interpretación:
# - Cada coeficiente mide el efecto parcial, controlando las demás variables
# - Verificar significancia (p-valores) y bondad de ajuste (R² ajustado)
# Diagnóstico visual de residuos
par(mfrow = c(2,2))
plot(modelo_multiple)
```

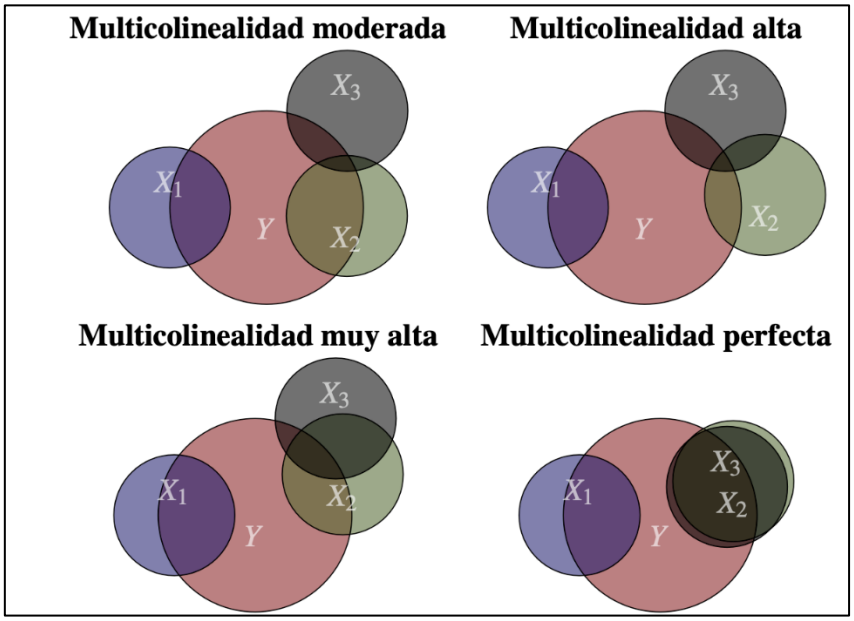


10.4. Multicolinealidad: detección (VIF)

La multicolinealidad es un problema que surge cuando dos o más variables explicativas de un modelo de regresión presentan una alta correlación entre sí. Esto dificulta identificar con precisión el efecto individual de cada variable sobre la variable dependiente, debido a que comparten información similar. Si bien no afecta necesariamente la capacidad predictiva global del modelo, sí puede comprometer la interpretación y confiabilidad de los coeficientes estimados.

Principales consecuencias

- **Coeficientes inestables:** pequeñas variaciones en la muestra pueden generar cambios importantes en los estimadores.
- **Errores estándar elevados:** reduce la precisión de las estimaciones.
- **Pérdida de significancia estadística:** variables relevantes pueden aparecer como no significativas.
- **Dificultad interpretativa:** se complica medir el efecto aislado de cada variable explicativa.



Métodos de detección

- **Matriz de correlación:** permite identificar relaciones altas entre variables independientes.
- **Factor de Inflación de la Varianza (VIF):** valores altos indican problemas de colinealidad; comúnmente un VIF mayor a 10 sugiere riesgo importante.
- **Signos inesperados o cambios bruscos en coeficientes:** pueden ser señales de multicolinealidad.

Posibles soluciones

- Eliminar una de las variables altamente correlacionadas.
- Combinar variables similares en un solo indicador.

- Aplicar transformaciones a las variables.
- Utilizar técnicas de reducción de dimensión, como componentes principales.
- Incrementar el tamaño de la muestra para mejorar la precisión de las estimaciones.

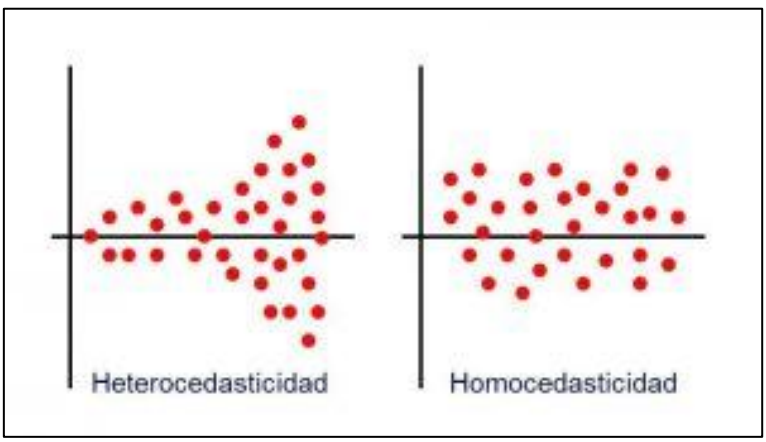
```
# Instalar paquetes necesarios
install.packages(c("car", "lmtest", "sandwich"))
library(car)
library(lmtest)
library(sandwich)
# Multicolinealidad
vif(modelo_multiple)   # VIF > 10 sugiere multicolinealidad alta
# Solución: eliminar variables redundantes o usar componentes principales
```

10.5. Heterocedasticidad: detección (Breusch-Pagan, White)

La heterocedasticidad es un problema econométrico que se presenta cuando la varianza del término de error no es constante para todos los valores de las variables explicativas. En otras palabras, la dispersión de los errores cambia a lo largo de las observaciones, incumpliendo el supuesto de homocedasticidad del modelo de regresión lineal clásico. Aunque este problema no sesga los coeficientes estimados por Mínimos Cuadrados Ordinarios, sí afecta la eficiencia de las estimaciones y la validez de las pruebas estadísticas.

Principales consecuencias

- **Errores estándar incorrectos**: afectan la precisión de las estimaciones.
- **Pruebas t y F poco confiables**: pueden generar conclusiones erróneas sobre significancia estadística.
- **Pérdida de eficiencia**: los estimadores dejan de ser los más eficientes dentro de su clase.
- **Inferencias engañosas**: riesgo de falsos positivos o falsos negativos en hipótesis.



Métodos de detección

- **Gráfico de residuos vs. valores ajustados**: permite observar patrones de dispersión no constante.
- **Prueba de Breusch-Pagan**: contrasta si la varianza de los errores depende de las variables explicativas.
- **Prueba de White**: detecta heterocedasticidad general sin asumir una forma funcional específica.
- **Análisis visual de residuos**: patrones en forma de abanico o embudo suelen indicar heterocedasticidad.

Posibles soluciones

- Utilizar errores estándar robustos para corregir la inferencia estadística.
- Transformar variables (por ejemplo, logaritmos).
- Reespecificar el modelo incorporando variables omitidas relevantes.
- Aplicar Mínimos Cuadrados Ponderados (WLS) cuando la estructura de varianza es conocida.
- Revisar valores atípicos o problemas de medición en los datos.

```
# Heterocedasticidad
  # Test de Breusch-Pagan
  bptest(modelo_multiple)
  # Test de White
  bptest(modelo_multiple, ~ fitted(modelo_multiple) + I(fitted(modelo_multiple)^2))
  # Corrección: errores robustos de White
  coeftest(modelo_multiple, vcov = vcovHC(modelo_multiple, type = "HC1"))
```

10.6. Autocorrelación: detección (Durbin-Watson)

La autocorrelación ocurre cuando los errores de un modelo de regresión están correlacionados entre sí a lo largo del tiempo o del orden de las observaciones. Es frecuente en datos de series temporales, donde los valores actuales pueden depender de periodos anteriores. Esta situación incumple el supuesto de independencia de los errores del modelo clásico y puede afectar la calidad de las inferencias estadísticas. Aunque los coeficientes estimados pueden seguir siendo insesgados bajo ciertos supuestos, dejan de ser eficientes.

Principales consecuencias

- **Errores estándar incorrectos:** afecta la precisión de las estimaciones.
- **Pruebas t y F poco confiables:** pueden generar conclusiones erróneas.
- **Pérdida de eficiencia:** los estimadores ya no son óptimos.
- **Sobreestimación del ajuste del modelo:** puede inflar indicadores como el R^2.
- **Problemas de predicción dinámica:** especialmente en series de tiempo.

Métodos de detección

- **Prueba de Durbin-Watson:** valores cercanos a 2 sugieren ausencia de autocorrelación; menores a 2 indican autocorrelación positiva y mayores a 2 negativa.
- **Prueba de Breusch-Godfrey:** útil para detectar autocorrelación de orden superior.
- **Gráfico de residuos en el tiempo:** permite observar patrones persistentes o cíclicos.
- **Correlograma de residuos:** identifica dependencia serial entre errores.

Posibles soluciones

- Incluir variables rezagadas o modelos dinámicos.
- Reespecificar el modelo incorporando variables omitidas relevantes.
- Transformar datos mediante diferencias o tasas de crecimiento.
- Utilizar errores estándar robustos a autocorrelación (Newey-West).
- Aplicar modelos especializados de series de tiempo, como ARIMA o regresiones dinámicas.

```
# Autocorrelación
# Creamos un ejemplo temporal
set.seed(123)
tiempo <- 1:30
serie <- 5 + 0.3 * tiempo + rnorm(30)
modelo_tiempo <- lm(serie ~ tiempo)
dwtest(modelo_tiempo)
# Si hay autocorrelación positiva (Durbin-Watson < 2)
# → se puede usar modelos con errores corregidos o modelos ARIMA.
```

10.7. Endogeneidad: detección (Hausman)

La endogeneidad es un problema econométrico que surge cuando una o más variables explicativas están correlacionadas con el término de error del modelo. Esto puede ocurrir por variables omitidas relevantes, simultaneidad entre variables, errores de medición o causalidad inversa. Cuando existe endogeneidad, se incumple el supuesto de exogeneidad, lo que provoca que los coeficientes estimados mediante Mínimos Cuadrados Ordinarios sean sesgados e inconsistentes. Por ello, representa una de las principales amenazas para la interpretación causal en los modelos de regresión.

Principales consecuencias

- **Estimadores sesgados e inconsistentes:** los coeficientes no reflejan el verdadero efecto económico.
- **Inferencia causal inválida:** dificulta identificar relaciones causa-efecto.
- **Resultados engañosos:** incluso con muestras grandes el problema persiste.

- **Decisiones incorrectas:** puede conducir a políticas o conclusiones erradas.
- **Mayor sensibilidad del modelo:** cambios en especificación alteran fuertemente los coeficientes.

Métodos de detección

- **Prueba de Hausman:** compara estimadores consistentes e inconsistentes para detectar endogeneidad.
- **Prueba Durbin-Wu-Hausman (DWH):** contrasta formalmente la exogeneidad de variables explicativas.
- **Evidencia teórica o institucional:** conocimiento del contexto económico puede sugerir simultaneidad o causalidad inversa.
- **Cambios fuertes en coeficientes:** al incluir nuevas variables de control puede indicar sesgo por omisión.
- **Análisis de residuos e instrumentos:** útil en modelos con variables instrumentales.

Posibles soluciones

- Utilizar **variables instrumentales (IV)** válidas y relevantes.
- Aplicar modelos de **efectos fijos** en datos panel para controlar heterogeneidad no observada.
- Emplear estimaciones en **diferencias o primeras diferencias**.
- Incorporar variables proxy cuando no se observan factores relevantes.
- Replantear la estrategia empírica con diseños cuasi experimentales o experimentales.

11. Modelos de elección discreta

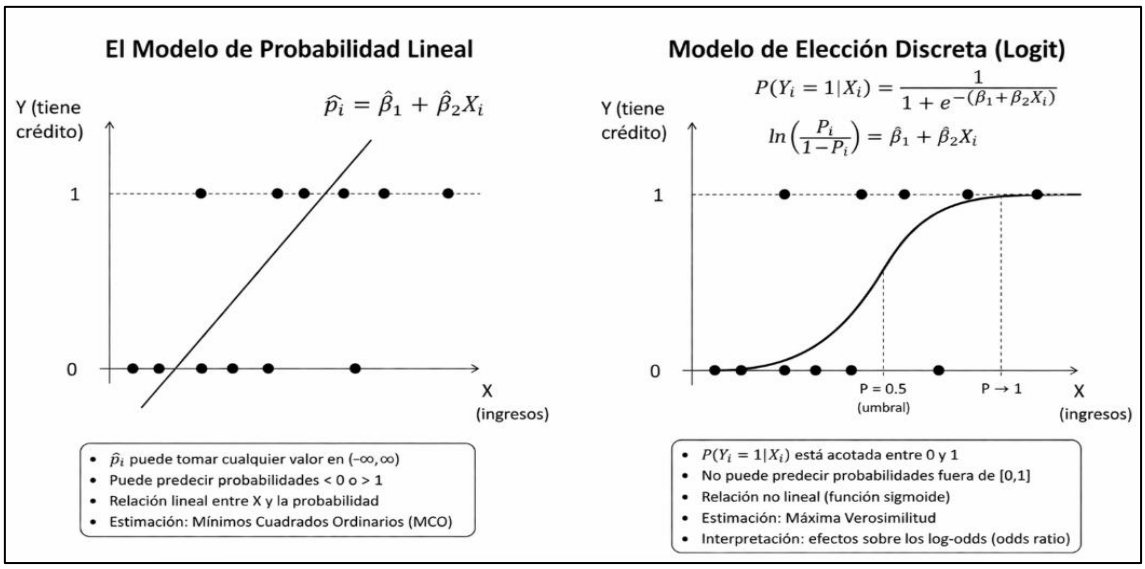
Los modelos de elección discreta se emplean cuando la variable dependiente representa decisiones o resultados cualitativos que toman un número limitado de categorías, en lugar de valores continuos. Son especialmente útiles cuando el interés analítico se centra en eventos binarios, como acceder o no al crédito, participar o no en el mercado laboral, adoptar o no una tecnología, o presentar o no una determinada condición de salud. En estos casos, los métodos de regresión lineal tradicional no resultan adecuados, ya que pueden generar predicciones fuera del rango lógico de probabilidades y no capturan correctamente la naturaleza no lineal de la decisión.

Entre los modelos más utilizados destacan el Logit y el Probit, los cuales estiman la probabilidad de ocurrencia de un evento en función de un conjunto de variables explicativas. Ambos permiten analizar cómo características individuales, económicas o sociales influyen en la probabilidad de elegir una determinada opción. La diferencia principal radica en la distribución estadística asumida para el término de error. La interpretación de estos modelos suele realizarse mediante efectos marginales, que indican cómo cambia la probabilidad estimada ante variaciones en las variables explicativas, manteniendo constantes los demás factores.

11.1. Diferencias con modelos de regresión lineal

A diferencia de la regresión lineal, estos modelos utilizan funciones no lineales para asegurar que las probabilidades estimadas se encuentren entre 0 y 1, ofreciendo así una representación más adecuada de decisiones binarias en el análisis económico. Se diferencian principalmente en la naturaleza de la variable dependiente y en la forma en que modelan la relación entre las variables.

Característica	Regresión Lineal	Modelo de elección discreta
Tipo de variable dependiente	Continua	Dicotómica (0/1)
Forma funcional	Lineal	No lineal (sigmoide)
Interpretación de coeficientes	Cambio en Y ante un cambio en X	Cambio en el logaritmo de los odds
Rango de predicción	(- ∞, + ∞)	(0, 1)
Ecuación	$y = \beta_0 + \beta_1 x_1 + \ ... \ + \beta_k x_k + u$	$P(y_i = 1 \mid x_i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ ... + \beta_k x_k)}}$
Método de estimación	Mínimos Cuadrados Ordinarios (MCO)	Máxima Verosimilitud
Problema principal	No aplica a variables binarias	Más complejo de interpretar



11.2. Estimación mediante modelos Logit y Probit

La principal diferencia entre ambos radica en la función de distribución que emplean, ya que el modelo Logit utiliza una distribución logística, mientras que el modelo Probit se basa en la distribución normal estándar. Es por ello que, la curva del Probit tiende a ser más suave en las colas, lo que refleja supuestos más estrictos sobre la distribución del término de error. En términos de estimación, ambos modelos suelen producir resultados muy similares, siendo el Logit el más utilizado en aplicaciones empíricas debido a su facilidad de interpretación y mayor robustez frente a valores extremos, mientras que el Probit es preferido en contextos más teóricos por su consistencia con supuestos de normalidad.

- **¿Qué son los odds ratios y efectos marginales?**

En los modelos Logit, la interpretación de los resultados se basa comúnmente en los odds y los odds ratios. Los odds representan la razón entre la probabilidad de que ocurra un evento y la probabilidad de que no ocurra. A partir de estos, los odds ratios indican cómo cambian esas probabilidades relativas cuando varía una variable explicativa. En particular, un odds ratio mayor a 1 implica que la variable incrementa la probabilidad del evento; un valor menor a 1 indica que la reduce; y un valor igual a 1 sugiere que no existe efecto. Esta medida es especialmente útil porque permite interpretar los coeficientes del modelo en términos relativos.

Odds	Odds Ratio
$odds = \frac{P(Y=1)}{1-P(Y=1)}$	$OR = \frac{odds\ con\ X + 1}{odds\ con\ X}$
• Si $OR > 1$ aumenta la probabilidad • Si $OR < 1$ disminuye la probabilidad • Si $OR = 1$ no hay efecto	

Por otro lado, los efectos marginales ofrecen una interpretación más directa en términos de probabilidad. Estos miden cuánto cambia la probabilidad de que ocurra el evento de interés (por ejemplo, Y=1) ante un pequeño cambio en una variable explicativa, manteniendo las demás constantes. A diferencia de los odds ratios, los efectos marginales se expresan en unidades de probabilidad (por ejemplo, puntos porcentuales), lo que facilita su comprensión e interpretación en aplicaciones empíricas.

$$\frac{\partial P(Y = 1 \mid X)}{\partial X_j}$$

```
# Modelos Logit y Probit
# Donde "x" significa logit o probit
modelo_x <- glm(informal ~ mujer + edad + educ + pobre,
                data = Base_Informalidad,
                family = binomial(link = "x"))
summary(modelo_x)
# Efectos marginales
summary(margins(modelo_x))
```

- **Ventajas y limitaciones**

	Logit	Probit
Ventajas	• Fácil interpretación mediante *odds ratios* • Mayor robustez a valores extremos • Computacionalmente sencillo	• Consistente con supuestos de normalidad • Mejor fundamentación teórica • Adecuado en modelos estructurales
Limitaciones	• Menor conexión con fundamentos teóricos • Supone distribución logística (no siempre realista)	• Interpretación menos intuitiva • Sensible a mala especificación • Requiere mayor esfuerzo para comunicar resultados

11.3. Interpretación de resultados (modelo Logit)

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.7718979  0.0638080  90.457   <2e-16 ***
mujer        0.2264069  0.0238306   9.501   <2e-16 ***
edad        -0.0212542  0.0007926 -26.814   <2e-16 ***
educ        -0.5768236  0.0059351 -97.188   <2e-16 ***
pobre        0.9649168  0.0414661  23.270   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 61632  on 58559  degrees of freedom
Residual deviance: 46038  on 58555  degrees of freedom
AIC: 46048

Number of Fisher Scoring iterations: 5

> exp(coef(modelo1_logit))
(Intercept)       mujer        edad        educ       pobre
321.1466740   1.2540859   0.9789701   0.5616796   2.6245694
> marg_modelo1 <- margins(modelo1_logit)
> summary(marg_modelo1)
 factor     AME     SE         z      p   lower   upper
   edad -0.0026 0.0001  -27.4469 0.0000 -0.0028 -0.0024
   educ -0.0713 0.0005 -144.6862 0.0000 -0.0723 -0.0704
  mujer  0.0280 0.0029    9.5139 0.0000  0.0222  0.0338
  pobre  0.1193 0.0051   23.4298 0.0000  0.1093  0.1293
```

- Coeficientes:

Ser mujer y ser pobre incrementan significativamente la probabilidad de ser informal, mientras que la edad y, especialmente, la educación la reducen. En términos de odds ratios, las mujeres tienen alrededor de 25% más probabilidades relativas de informalidad y las personas pobres más del 160%, mientras que cada año adicional de edad reduce esta probabilidad en aproximadamente 2% y mayores niveles educativos la disminuyen de forma importante (≈44%). Todos los efectos son estadísticamente significativos ($p < 0.01$).

- Efectos marginales:

- Edad: Un año adicional reduce la probabilidad de informalidad en 0.26 puntos porcentuales.
- Educación: Incrementos en educación reducen la probabilidad en 7.13 puntos porcentuales.
- Mujer: Ser mujer aumenta la probabilidad en 2.8 puntos porcentuales.
- Pobre: Ser pobre aumenta la probabilidad en 11.93 puntos porcentuales.
- Todos los efectos son estadísticamente significativos.

12. Evaluación de Impacto

En términos generales, una evaluación es un proceso sistemático y objetivo mediante el cual se analizan los resultados, efectos y desempeño de una política pública, programa o intervención. Su finalidad es determinar en qué medida una acción logró los objetivos planteados, identificar fortalezas y debilidades, y generar aprendizajes para futuras decisiones.

La evaluación de impacto es una herramienta clave en economía aplicada y en el diseño de políticas públicas, pues permite responder preguntas relevantes como: ¿cómo aumentar la productividad de los agricultores?, ¿cómo reducir la anemia infantil?, ¿cómo incrementar la asistencia escolar?, o ¿cómo mejorar el acceso al empleo formal? A través de metodologías rigurosas, este enfoque permite determinar si un programa realmente produjo mejoras en la población objetivo y en qué magnitud.

Asimismo, la evaluación de impacto contribuye a mejorar el diseño y funcionamiento de los programas, optimizar el uso de recursos públicos y fortalecer la rendición de cuentas. La evidencia generada permite identificar qué intervenciones funcionan, cuáles requieren ajustes y cuáles no generan los resultados esperados. De esta manera, se promueve una gestión pública basada en evidencia y una asignación más eficiente del presupuesto.

Conceptos clave de la evaluación de impacto

El impacto se define como la diferencia entre dos escenarios posibles para una misma persona, hogar o unidad de análisis:

- **Factual:** lo que realmente ocurre con los individuos que participan en el programa o reciben la intervención.
- **Contrafactual:** lo que habría ocurrido con esos mismos individuos si no hubieran participado en el programa.

El principal desafío metodológico consiste en estimar adecuadamente el contrafactual, ya que no es posible observar simultáneamente ambos escenarios para una misma unidad. En la práctica, esto se resuelve construyendo un grupo de comparación o grupo de control que sea lo más similar posible al grupo tratado en características observables y no observables relevantes. Si ambos grupos son comparables, las diferencias observadas en los resultados pueden interpretarse como el efecto atribuible al programa.

¿Qué es el contrafactual?

- Teórico: Es la construcción de un grupo similar al grupo de individuos que SI participa en el programa.
- Práctica: Es la selección de un grupo de individuos que no participa en el programa, este será el grupo de comparación o grupo de control.

12.1. Métodos de evaluación de impacto

La principal diferencia entre los métodos de evaluación de impacto radica en la forma en que se construye el contrafactual, es decir, el escenario que permite estimar qué habría ocurrido en ausencia de la intervención. En este sentido, los métodos experimentales son los más rigurosos, ya que asignan aleatoriamente a los beneficiarios entre un grupo tratado y un grupo de control, garantizando comparabilidad y reduciendo sesgos de selección. Por su parte, los métodos cuasiexperimentales buscan aproximarse a esta lógica cuando la aleatorización no es posible, utilizando técnicas estadísticas como diferencias en diferencias, regresión discontinua o emparejamiento para construir un grupo de comparación creíble.

En contraste, los métodos no experimentales no construyen un contrafactual explícito o lo hacen de manera más limitada, basándose en comparaciones simples o modelos econométricos sin un diseño robusto de identificación causal. Si bien pueden ser útiles para análisis descriptivos o exploratorios, presentan mayores riesgos de sesgo y menor validez para atribuir causalidad. En conjunto, la elección del método depende de la disponibilidad de datos, el contexto de implementación y el nivel de rigurosidad requerido para la toma de decisiones.

12.1.1. Métodos no experimentales: Antes y después

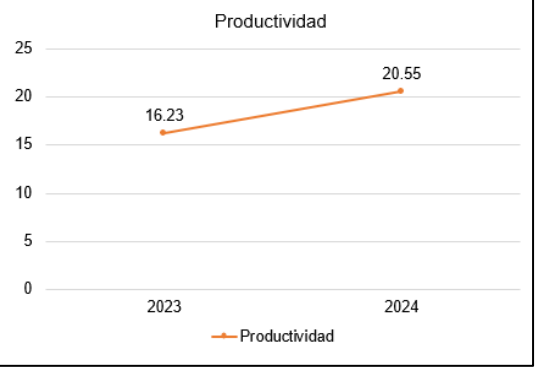
El método de antes y después estima el impacto de una intervención comparando los resultados de los mismos individuos en dos momentos del tiempo: previo y posterior a la implementación del programa. Bajo este enfoque, la variación observada se interpreta como el efecto del programa, ya que se asume que los cambios en el indicador de interés reflejan la influencia directa de la intervención.

Sin embargo, este método descansa en un supuesto fuerte: que no han ocurrido otros factores relevantes entre ambos periodos que puedan afectar el resultado. En la práctica, esto implica asumir la ausencia de tendencias preexistentes, shocks externos o cambios en el entorno que influyan en la variable analizada. Por ello, aunque es un enfoque sencillo y fácil de aplicar, su capacidad para identificar efectos causales es limitada, ya que puede atribuir erróneamente al programa cambios que en realidad responden a otros factores.

$$Impacto = Y^{T}_{t=1} - Y^{T}_{t=0}$$

Antes y Después

Productividad	Al final del programa	20.55
	Al inicio del programa	16.23
	Diferencia	**4.23**



Ventajas	Limitaciones
Fácil de aplicar	No controla factores externos
Requiere pocos datos	Puede generar estimaciones sesgadas
Útil como análisis inicial	Confunde tendencia con impacto

12.1.2. Métodos no experimentales: Diferencia simple

El método de diferencia simple estima el impacto de una intervención comparando los resultados entre un grupo que recibe el tratamiento y otro que no lo recibe en un mismo momento del tiempo. Bajo este enfoque, la diferencia observada entre ambos grupos se interpreta como el efecto del programa, asumiendo que dicha brecha refleja exclusivamente la influencia de la intervención.

No obstante, este método se basa en un supuesto exigente: que ambos grupos son comparables desde el inicio, es decir, que no existen diferencias sistemáticas entre ellos más allá de la participación en el programa. En la práctica, esta condición rara vez se cumple, ya que pueden existir sesgos de selección o características no observadas que influyan en los resultados. Por ello, aunque es un enfoque sencillo y de fácil implementación, su capacidad para identificar efectos causales es limitada si no se cuenta con grupos realmente comparables.

$$Impacto = Y_{t=1}^{T} - Y_{t=1}^{C}$$

Diferencia Simple

Productividad	Agricultores con el programa	21.23
	Agricultores sin el programa	17.23
	Diferencia	**4.0**

Ventajas	Limitaciones
Fácil interpretación	Sesgo de selección
Comparación directa	Grupos pueden no ser comparables
Requiere datos simples	No controla diferencias iniciales

12.1.3. Métodos cuasiexperimentales: Diferencias en diferencias

El método de diferencias en diferencias (DiD) estima el impacto de una intervención comparando cómo cambian los resultados en el tiempo entre un grupo tratado y un grupo de control. A diferencia de los enfoques más simples, este método no solo observa niveles en un momento específico, sino que analiza la evolución antes y después de la intervención en ambos grupos. El efecto del programa se obtiene como la diferencia entre estos cambios, lo que permite aislar, en cierta medida, factores comunes que afectan a ambos grupos en el tiempo.

Este enfoque se sustenta en el supuesto de tendencias paralelas, el cual establece que, en ausencia del programa, ambos grupos habrían seguido trayectorias similares en el tiempo. Si este supuesto se cumple, las diferencias observadas después de la intervención pueden atribuirse al programa. Sin embargo, si los grupos ya presentaban dinámicas distintas antes de la intervención, las estimaciones pueden estar sesgadas, por lo que es fundamental verificar empíricamente la validez de este supuesto antes de interpretar los resultados.

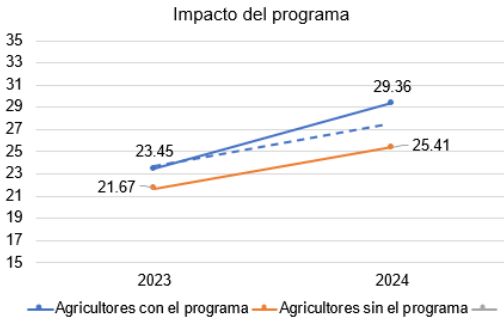
$$DiD = Impacto_T - Impacto_C$$

$$ATE = (Y_1^T - Y_0^T) - (Y_1^C - Y_0^C)$$

Diferencias en Diferencias

$$DiD = (Y_1^T - Y_0^T) - (Y_1^C - Y_0^C)$$

	Grupos	2023	2024	Dif
Productividad	Agricultores con el programa	23.45	29.36	5.91
	Agricultores sin el programa	21.67	25.41	3.74
	Diferencia			2.17



En el método de diferencias en diferencias (DiD), el Efecto Promedio del Tratamiento (ATE) se interpreta a partir del valor estimado del coeficiente de interacción entre tratamiento y tiempo. Si el valor es mayor a cero, indica que el programa tuvo un efecto positivo sobre el resultado; si es igual a cero, sugiere que no hubo impacto; y si es menor a cero, implica un efecto negativo.

Ventajas	Limitaciones
Controla diferencias iniciales	Requiere tendencias paralelas
Considera cambios en el tiempo	Sensible a shocks distintos
Muy usado en políticas públicas	Necesita datos panel o repetidos

```
# Calculo de average treatment effect (ATE)
modelo <- lm(ingreso ~ tratamiento*post, data = data_dif)
summary(modelo)
```

12.1.4. Métodos cuasiexperimentales: Propensity Score Matching

El método de Propensity Score Matching (PSM) busca estimar el impacto de una intervención construyendo un grupo de control comparable a partir de individuos no tratados que presentan características similares a los beneficiarios. Para ello, primero se estima la probabilidad de participar en el programa —conocida como propensity score— en función de variables observables como edad, nivel educativo, ingresos u otras características relevantes. Posteriormente, cada individuo tratado es emparejado con uno o varios individuos no tratados con probabilidades similares, permitiendo así realizar comparaciones más equilibradas entre ambos grupos.

Este enfoque se basa en el supuesto de independencia condicional, que establece que, una vez controladas las variables observables, no existen factores no observados que influyan simultáneamente en la participación en el programa y en los resultados. En consecuencia, el PSM corrige únicamente el sesgo de selección asociado a variables observables, por lo que su validez depende críticamente de la calidad, disponibilidad y pertinencia de la información utilizada. Si existen factores no observados relevantes que no han sido considerados, las estimaciones pueden seguir estando sesgadas.

$$P(X) = P(D = 1 \mid X)$$

$$ATT = E[Y(1) - Y(0) \mid D = 1]$$

Ventajas	Limitaciones
Mejora comparabilidad	No controla no observables
Reduce sesgo observable	Depende de calidad de datos
Intuitivo conceptualmente	Sensible a mala especificación

```
# Base
data("lalonde")
table(lalonde$treat)
prop1<-matchit(treat ~ age + educ + race + married + nodegree, data=lalonde, method = "nearest", distance = "glm")
lalondem<-match.data(prop1)
table(lalondem$treat)
t.test(re78 ~ treat, data = lalondem)
```

```
        Welch Two Sample t-test

data:  re78 by treat
t = -0.66533, df = 353.47, p-value = 0.5063
alternative hypothesis: true difference in means between group 0 and group 1 is not e
qual to 0
95 percent confidence interval:
 -1963.1925   970.6725
sample estimates:
mean in group 0 mean in group 1
       5852.884        6349.144
```

12.1.5. Métodos cuasiexperimentales: Regresión discontinua

El método de regresión discontinua (RDD) identifica el impacto de una intervención aprovechando la existencia de un punto de corte en una variable de asignación, a partir del cual se determina quién accede al programa. La lógica consiste en comparar individuos ubicados muy cerca de ese umbral, ya que comparten características muy similares, con la única diferencia de que algunos reciben el tratamiento y otros no. Esta comparación local permite aproximarse a un experimento natural, generando estimaciones con alta validez interna en el entorno cercano al punto de corte.

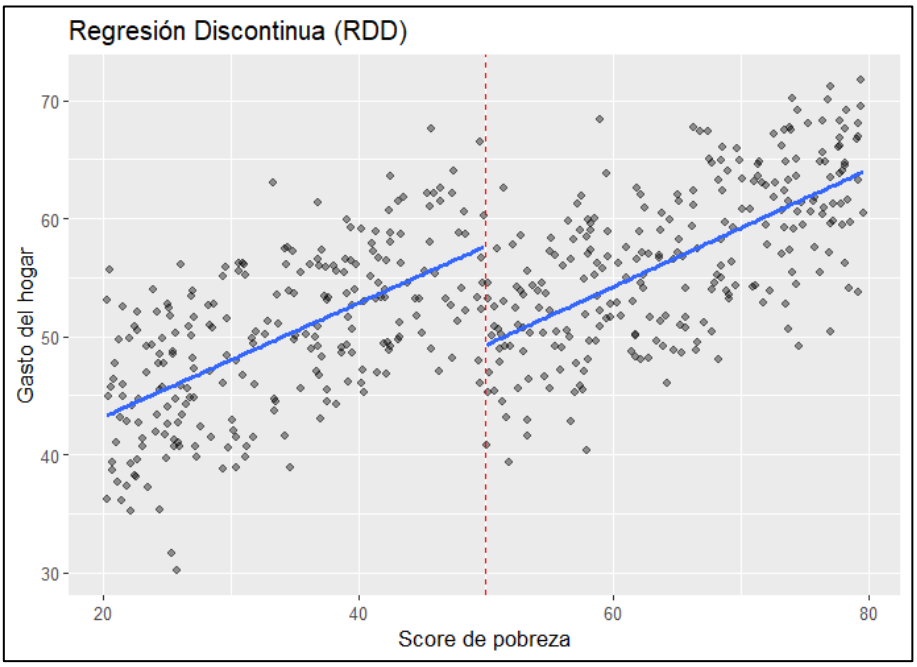
El supuesto fundamental del RDD es que, en ausencia del programa, la relación entre la variable de resultado y la variable de asignación sería continua en torno al umbral. Por tanto, cualquier salto o discontinuidad observada en ese punto puede interpretarse como el efecto causal de la intervención. Asimismo, se requiere que los individuos no puedan manipular estratégicamente su posición respecto al umbral, ya que ello rompería la comparabilidad entre los grupos y sesgaría los resultados. Por esta razón, es fundamental verificar empíricamente la continuidad de las variables y la ausencia de manipulación en torno al punto de

$$Y_i = \alpha + (\beta X_i - c) + \rho D_i + \varepsilon$$

- ✓ ρ: Parámetro de interés
- ✓ c: Punto de corte
- ✓ X_i: Índice de elegibilidad
- ✓ D_i: Tratamiento (1 si cumple el umbral)

Ventajas	Limitaciones
Alta validez causal	Solo efecto local
No requiere aleatorización	Requiere datos cerca del umbral
Intuitivo gráficamente	Sensible a manipulación del corte

```
# Gráfico RDD
ggplot(data_RD, aes(x = score_pobreza, y = resultado_gasto)) +
  geom_point(alpha = 0.4) +
  geom_vline(xintercept = 50, linetype = "dashed", color = "red") +
  geom_smooth(data = subset(data_RD, score_pobreza <= 50), method = "lm", se = FALSE) +
  geom_smooth(data = subset(data_RD, score_pobreza > 50), method = "lm", se = FALSE) +
  labs(title = "Regresión Discontinua (RDD)",
       x = "Score de pobreza",
       y = "Gasto del hogar")
# Crear running variable
data_RD$running <- data_RD$score_pobreza - 50
# Modelo extendido
model_2 <- lm(resultado_gasto ~ tratamiento + running + sexo + edad + zona, data = data_RD)
summary(model_2)
```



12.1.6. Métodos cuasiexperimentales: Variable Instrumental

El método de variables instrumentales (VI) se emplea cuando existe endogeneidad, es decir, cuando la variable de tratamiento está correlacionada con factores no observados que también influyen en el resultado. Para enfrentar este problema, se introduce un instrumento: una variable externa que afecta la probabilidad de participar en el programa, pero que no tiene un efecto directo sobre la variable de resultado. Así, el instrumento permite aislar una fuente de variación exógena que hace posible identificar efectos causales más confiables.

La implementación del método se realiza en dos etapas. En la primera etapa, el instrumento se utiliza para explicar la asignación al tratamiento, estimando qué parte de la participación en el programa puede atribuirse a factores exógenos y no a decisiones individuales o variables omitidas. Este paso permite depurar la variable de tratamiento de la parte que está contaminada por endogeneidad.

$$D_i = \pi_0 + \pi_1 Z_i + \pi_2 X_i + u_i$$

En la segunda etapa, se utiliza la variable de tratamiento predicha en la primera etapa para estimar su efecto sobre el resultado de interés. De esta manera, el método logra aislar la variación exógena del tratamiento y corregir el sesgo por endogeneidad. No obstante, su validez depende de que el instrumento cumpla dos condiciones clave: relevancia, es decir, que esté correlacionado con el tratamiento; y exogeneidad, que garantice que no esté relacionado con el término de error ni afecte directamente el resultado.

$$Y_i = \beta_0 + \beta_1 \widehat{D}_i + \beta_2 X_i + \varepsilon_i$$

Ventajas	Limitaciones
Corrige endogeneidad	Difícil encontrar instrumentos
Permite estimar causalidad	Interpretación limitada (LATE)
Útil en contextos complejos	Sensible a mala elección del instrumento

```
# Modelo IV (2SLS)
model_iv <- ivreg(resultado_perdidas_ev ~ transferencia + riesgo_indice + poblacion + ingreso_pc |
                    instrumento_lluvia + riesgo_indice + poblacion + ingreso_pc,
                  data = data_VI)
summary(model_iv)
# Test de endogeneidad + diagnósticos
summary(model_iv, diagnostics = TRUE)
```

12.1.7. Métodos experimentales: Ensayos controlados aleatorizados

Los ensayos controlados aleatorizados (RCT) consisten en asignar el tratamiento de forma aleatoria entre los participantes elegibles, de modo que el grupo tratado y el grupo de control sean, en promedio, equivalentes en sus características observables y no observables. Esta asignación al azar elimina el sesgo de selección y asegura que cualquier diferencia en los resultados posterior a la intervención pueda atribuirse directamente al programa. Por esta razón, los RCT son considerados el estándar más riguroso para la identificación de efectos causales en la evaluación de impacto.

El supuesto central de este enfoque es que la aleatorización se implementa correctamente y se mantiene durante todo el estudio, garantizando la comparabilidad entre los grupos. Asimismo, es importante que no existan problemas como el incumplimiento del tratamiento (cuando los asignados no reciben la intervención) o el abandono diferencial entre grupos, ya que estos pueden introducir sesgos y afectar la validez de las estimaciones. Por ello, el diseño y la implementación del experimento deben ser cuidadosamente controlados para preservar la integridad de los resultados.

Ventajas	Limitaciones
Alta validez interna	Costoso de implementar
Identificación causal directa	Puede ser poco ético en algunos casos
Fácil interpretación	Limitada validez externa

12.2. Interpretación de resultados por método de evaluación de impacto

Método	¿Cómo se interpreta?	¿Qué identifica realmente?
Antes y después	Cambio en el tiempo del mismo grupo	Variación temporal observada tras la intervención
Diferencia simple	Diferencia entre tratados y no tratados	Brecha promedio entre ambos grupos en un momento dado
Diferencias en diferencias (DiD)	Cambio diferencial entre grupos en el tiempo	Efecto neto descontando tendencias comunes
Propensity Score Matching (PSM)	Diferencia entre tratados y comparables emparejados	Efecto promedio entre individuos comparables en variables observables
Regresión Discontinua (RDD)	Diferencia en el punto de corte	Efecto local del tratamiento en torno al umbral
Variable Instrumental (VI)	Efecto inducido por el instrumento	Efecto causal local (LATE) en la población afectada por el instrumento
Ensayos aleatorizados (RCT)	Diferencia promedio entre grupos asignados aleatoriamente	Efecto causal promedio del tratamiento

La evaluación de impacto no se limita a describir cambios en los resultados, sino que busca identificar efectos causales atribuibles directamente a una intervención. Esto implica distinguir entre lo que ocurrió como consecuencia del programa y lo que habría sucedido de todas formas en ausencia del mismo. En este contexto, el concepto de contrafactual es central, ya que permite establecer un punto de comparación creíble para aislar el verdadero efecto del tratamiento.

La elección del método de evaluación depende no solo de la disponibilidad y calidad de los datos, sino, principalmente, de la capacidad para construir un contrafactual válido y comparable. Métodos más rigurosos logran aproximarse mejor a este escenario ideal, reduciendo sesgos y fortaleciendo la validez de las conclusiones. Sin embargo, en la práctica, la selección metodológica también debe considerar las restricciones operativas, el contexto de implementación y los supuestos que pueden sostenerse empíricamente. En consecuencia, una buena evaluación no es necesariamente la más compleja, sino aquella que utiliza el método más adecuado y creíble para responder la pregunta de investigación planteada.

13. Bibliografía

- Joshi, A. P., & Patel, B. V. (2020). Data preprocessing: The techniques for preparing clean and quality data for data analytics process. Oriental Journal of Computer Science and Technology, 13(2–3), 78–81. http://dx.doi.org/10.13005/ojcst13.0203.03
- Koukaras, P., & Tjortjis, C. (2025). Data preprocessing and feature engineering for data mining: Techniques, tools, and best practices. AI, 6(10), 257. https://doi.org/10.3390/ai6100257
- Wooldridge, J. M. (2010). Introducción a la econometría: Un enfoque moderno (4.ª ed.). Thomson Learning. Disponible.
- Roberto Hernández Sampieri, Carlos Fernández Collado, & Pilar Baptista Lucio (2014). Metodología de la investigación (6.ª ed.). McGraw-Hill.
- Rufino E. Moya Calderón (2016). Estadística descriptiva e inferencial. Editorial San Marcos.
- Damodar N. Gujarati, D. N., & Dawn C. Porter (2010). Econometría (5.ª ed.). McGraw-Hill. Disponible.

Repositorio:

- Curso de R de La Universidad Complutense de Madrid

  Disponible en: 1346-2019-05-04-Curso de R.pdf

- Guía de RStudio por Jordi Mas Elias de Universitat Oberta de Catalunya

  Disponible en:

  https://openaccess.uoc.edu/server/api/core/bitstreams/871e35eb-8c02-435d-a7cd-5652aebc5aff/content

- Manual de R por Freddy Hernández y Olga Usuga - 2024

  Disponible en: https://fhernanb.github.io/Manual-de-R/

- Introducción al uso de R y R Commander para el análisis estadístico de datos en ciencias sociales por Rosario Collatón - 2014

  Disponible en: https://cran.r-project.org/doc/contrib/Chicana-Introduccion_al_uso_de_R.pdf