
Universidad de Valladolid

# Facultad de Ciencias Económicas y Empresariales

## Trabajo Fin de Grado

## Grado en Administración y Dirección de Empresas

# Econometría Básica con R

Presentado por:

***Diego Gómez Gonzalo***

Tutelado por:

***César García Gómez***

*Valladolid, 16 de junio de 2025*

**Resumen**

En este trabajo se presenta una introducción al análisis econométrico con el lenguaje de programación R, demostrando sus funcionalidades para llevar a cabo dicho análisis. Para ello, se introduce brevemente los orígenes de R, al igual que sus ventajas y limitaciones respecto a otros programas estadísticos. Posteriormente, veremos el procedimiento de instalación de R y RStudio, y exploraremos los diferentes tipos de estructuras de datos en R. Una vez comprendida la interfaz y su funcionamiento básico, se presentarán los principales comandos para realizar estadística descriptiva básica y análisis de regresión lineal múltiple, estudiando los principales aspectos econométricos para la validación del modelo. El objetivo principal del trabajo es ofrecer al lector un acercamiento al lenguaje R mediante un ejemplo práctico para poder realizar análisis estadísticos y econométricos aprovechando sus herramientas, amplias funcionalidades y su disponibilidad gratuita.

**Palabras clave**: R, RStudio, análisis econométrico, regresión lineal múltiple.

**Abstract**

This work presents an introduction to econometric analysis with the R programming language, demonstrating its functionalities to carry out this type of analysis. To do so, we briefly introduce the origins of R, as well as its advantages and limitations with respect to other statistical programs. Subsequently, we will see the installation procedure of R and RStudio, and we will explore the different types of data structures in R. Once the interface and its basic operation are understood, the main commands to perform basic descriptive statistics and multiple linear regression analysis will be presented, studying the main econometric aspects for the validation of the model. The main objective of the work is to offer the reader an approach to the R language by means of a practical example to be able to carry out statistical and econometric analyses taking advantage of its tools, extensive functionalities and its free availability.

**Key words**: R, RStudio, econometric analysis, multiple linear regression.

# ÍNDICE

**ÍNDICE DE FIGURAS**



# 1 INTRODUCCIÓN

En el campo de la estadística, ya sea bioestadística, econometría, machine learning, análisis de datos etc., se ha vuelto imprescindible el uso de software para la manipulación de datos, creación de modelos predictivos, representación de datos y automatización de cálculos, entre otras funciones.

En las últimas décadas, R se ha convertido en uno de los lenguajes de programación más populares debido a su gran utilidad para el análisis estadístico y la creación de gráficos. Esta popularidad se ha visto impulsada por su gran ecosistema, la creciente comunidad de usuarios, el aumento de paquetes disponibles y el aumento de materiales académicos, según Verzani (2014). Además, R destaca por la gran variedad de técnicas estadísticas que ofrece (modelado lineal y no lineal, pruebas estadísticas clásicas, análisis de series temporales, clasificación, análisis clúster, etc.).

R es un lenguaje y entorno de software libre y de código abierto bajo los términos de la Licencia Pública General de GNU (GPL), siendo su principal objetivo proporcionar a los usuarios libertad de usar, estudiar, compartir y modificar el software al mismo tiempo que protegerlo ante intentos de apropiación. Está disponible en varias plataformas como Linux, Windows y MacOS.

Para entender su origen, hay que hablar del lenguaje S, del que R deriva y con el que comparte gran parte de su estructura. S fue desarrollado por John Chambers y otros

expertos como Richard Becker y Allan Wilks en los Laboratorios Bell de AT&T. Fue lanzado en 1976 como un entorno interno de análisis de datos, siendo sus primeras versiones bastante básicas. Sin embargo, actualmente es un lenguaje de programación completo, disponible como S-PLUS, comercializado por TIBCO.

El desarrollo de R comenzó en 1992 en la Universidad de Auckland (Nueva Zelanda) por Robert Gentleman y Ross Ihaka. Su objetivo inicial era crear un lenguaje didáctico para utilizar en sus clases de estadística en la universidad, combinando las fortalezas de los lenguajes S y Scheme. El nombre de R proviene de las iniciales de los creadores y de su relación con S. El código fuente fue liberado por la (GPL) en 1995 y desde 1997 está a cargo del R Core Team, un grupo de personas que se encargan del mantenimiento y desarrollo.

R es un lenguaje orientado a objetos y de tipo interpretado, es decir, que trabaja con datos de forma organizada y estructurada y que no necesita ser compilado antes de su ejecución. Una de sus mayores ventajas es la gran disponibilidad de paquetes, los cuales son colecciones de funciones y conjuntos de datos desarrollados por la comunidad para facilitar el uso a los usuarios. Estos paquetes se encuentran disponibles en CRAN, que es el repositorio oficial de R (https://cran.r-project.org/web/packages/available_packages_by_name.html)

Actualmente existen varios softwares dedicados al análisis estadístico, como SPSS, EViews, Stata o Minitab, los cuales presentan unas interfaces gráficas muy intuitivas basadas en menús y ventanas. Esto reduce notablemente la curva de aprendizaje, convirtiéndolo en la mejor opción para usuarios sin experiencia en programación. Sin embargo, tienen grandes desventajas, como unas licencias muy costosas y el decreciente número de usuarios, lo que conduce a menos soporte y foros. Por otra parte, R es un lenguaje de programación, dando lugar a una mayor dificultad de aprendizaje, pero a su vez ofrece una mayor flexibilidad, capacidad de modificar el código y añadir paquetes según las necesidades del usuario.

En cuanto a la organización del trabajo, se ha estructurado de la siguiente manera. La segunda sección muestra el proceso de instalación de R y RStudio en nuestro equipo, cómo desenvolvernos por la interfaz y la presentación de los diferentes tipos de

estructuras de datos en R (vectores, matrices, listas y data frames). En la tercera sección se muestran los comandos para realizar análisis descriptivos básicos, al igual que la creación de gráficos sencillos con el paquete ggplot. En la cuarta sección se abordan los procedimientos esenciales para llevar a cabo un análisis de regresión lineal múltiple, cubriendo desde la estimación del modelo hasta la validación de este mediante un ejemplo práctico. Finalmente, en la sección 6 se exponen las principales conclusiones del trabajo.

# 2 INSTALACION DE R Y RSRUDIO E INTRODUCCION DE DATOS

## 2.1 Instalación de R

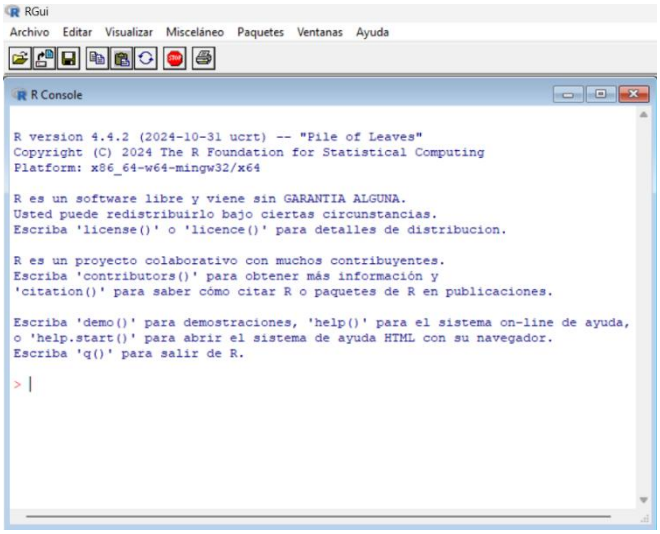
El proceso de instalación de R es muy sencillo, simplemente hay que seguir las siguientes instrucciones de la página CRAN (Comprehensive R Archive Network):

- Nos dirigimos a la siguiente página web https://cran.r-project.org/. Una vez en la página, pinchamos en “Descargar e Instalar R” en nuestro respectivo sistema operativo (Windows, MacOS o Linux). Posteriormente seleccionaremos en “install R for the first time” y descargaremos R-4.4.2.
- Una vez descargado en nuestro disco duro R-4.4.2-win.exe, debemos ejecutar el programa y aceptar que haga cambios en el equipo. Posteriormente, nos aparecerá una ventana emergente en la que se puede configurar diferentes opciones, como el idioma y el directorio donde queremos instalar R. Luego, vamos aceptando las demás opciones preseleccionadas.

## 2.2 Interfaz de R

**Figura 1**

*Ventana principal de R*



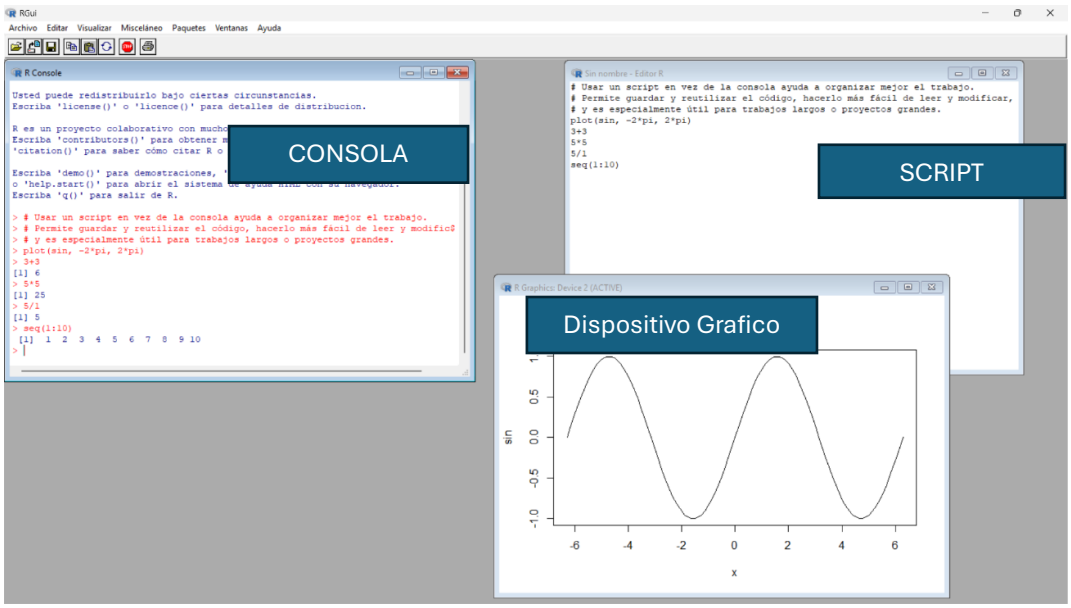

***Nota.*** Elaboración propia

Como podemos ver en la imagen, al abrir R se muestra RGui, su interfaz gráfica por defecto. En la parte superior encontramos la barra de tareas y herramientas.

En la siguiente imagen he realizado un pequeño ejemplo para ver las tres ventanas principales de R.

**Figura 2**

*Las tres principales ventanas de R*




***Nota***. Elaboración propia.

Como mencionamos anteriormente, la mayor desventaja que tiene R frente a otros softwares estadísticos es la complejidad de la interfaz para nuevos usuarios sin experiencia en programación. Por ello han surgido diferentes entornos de desarrollo integrado (IDE) e interfaces gráficas de usuario (GUI) gratuitos, como RStudio y Rcomander, los cuales hacen que trabajar con R sea más intuitivo, ordenado y visual. En nuestro caso vamos a utilizar RStudio.

### 2.3 Instalación de RStudio

El proceso de instalación de Rstudio es muy sencillo, únicamente deberemos seguir las instrucciones de la siguiente página web Posit, el nuevo nombre de la página oficial de Rstudio.

- Nos dirigimos al siguiente link https://posit.co/download/rstudio-desktop/. Una vez en la página, pinchamos donde pone descargar Rstudio en nuestro respectivo sistema operativo.
- Una vez descargado ejecutamos el programa aceptando todas las opciones preseleccionadas.

### 2.4 Interfaz de RStudio

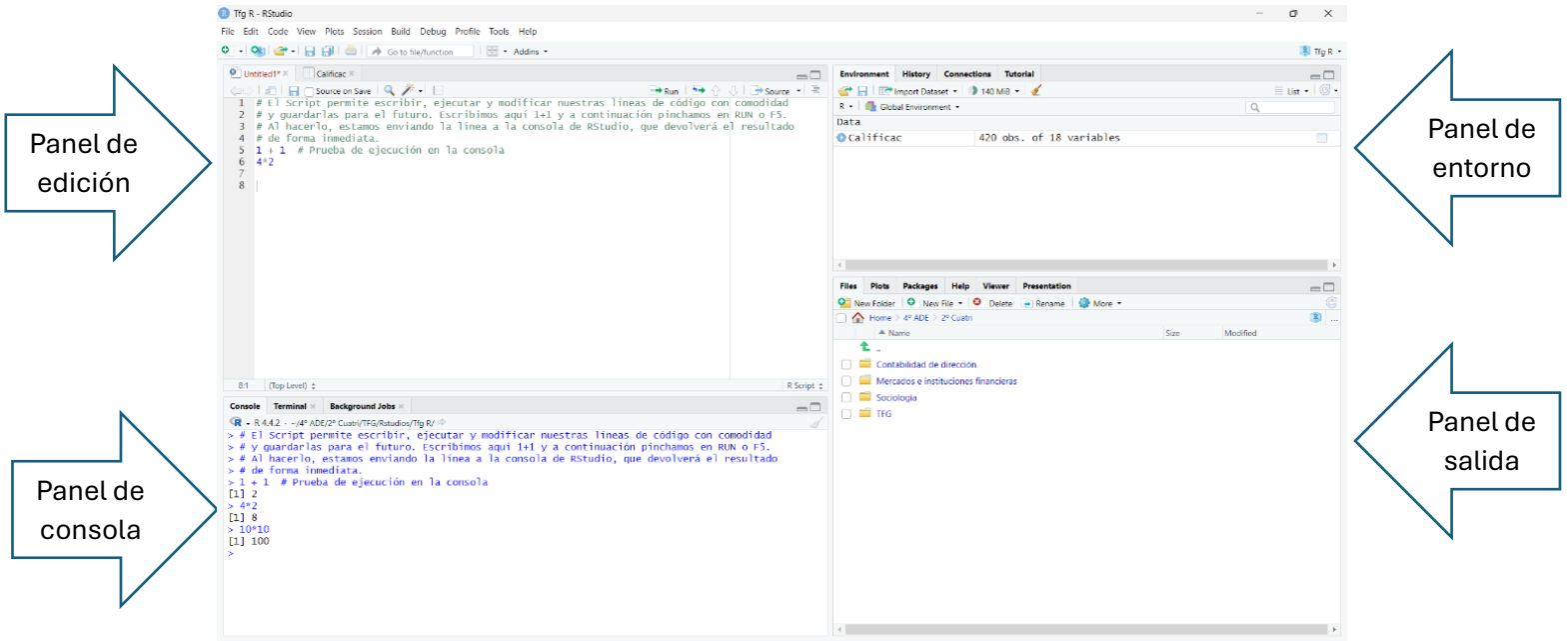
Una vez abierto RStudio, veremos 4 paneles principales en la interfaz: Source pane, Console pane, Environment pane y Output pane, como se muestra en la figura 3.

- El panel de consola se ubica abajo a la izquierda. Se utiliza para introducir y ejecutar comandos de manera directa, ya que vemos el resultado de manera inmediata. Las instrucciones y comandos se escriben inmediatamente después de “>” ejecutándose una por una. En R, se puede escribir código directamente desde la consola o desde scripts, como veremos a continuación. Además, mediante paquetes puede funcionar como una consola de Python. También en el panel encontramos las pestañas Terminal y Background para otras funcionalidades.
- El editor de código (Source pane) es donde puedes ver y editar los scripts. Se utiliza principalmente para guardar scripts de R, pero también para otros formatos como Python, archivos de texto, de CSS y para crear documentos computacionales como Quarto y R Markdown. Trabajar con scripts tiene la ventaja de poder guardar colecciones de comandos como archivo, facilitando la reutilización de los comandos en otra sesión sin la necesidad de volver a escribirlos.
- El panel de entorno se ubica arriba a la derecha, incluyendo las pestañas de Environment, History, Connections y Tutorial. Su funcionalidad es darnos la información de todos los objetos y variables que tengamos definidos en nuestro proyecto, así como permitirnos guardar el espacio de trabajo e importar datos. La pestaña de History nos permite visualizar y guardar los comandos ejecutados durante la sesión.
- El panel de salida se ubica en la parte inferior derecha, incluyendo las pestañas de files, plots, packages, help, viewer y presentación. En files podemos ver y gestionar los archivos del ordenador en el que tenemos nuestro proyecto en R guardado. En plots podemos ver todos los gráficos que hayamos ejecutado, así como hacer zoom y guardarlos. La pestaña de packages permite gestionar los paquetes en R, instalar nuevos paquetes, actualizar los existentes y cargar o

desactivar los paquetes en la sesión actual. Y por último la pestaña de help nos muestra documentos y recursos sobre R y Rstudio.

**Figura 3**

*Paneles principales de RStudio.*




***Nota***. Elaboración propia

### 2.5 Introducción de datos

Como hemos comentado en la introducción, R es un programa orientado a objetos. Esto quiere decir que las variables, datos, funciones, resultados, etc, se guardan en la memoria activa del computador en forma de objetos con un nombre especifico (Paradis, 2003), pudiendo ser modificados o manipulados por operadores y funciones.

Los objetos suelen tener diferentes atributos como:

- Nombre: Es la etiqueta que se le da al objeto.
- Tipo: Muestra si el objeto es numérico, de carácter o lógico. Para obtener el tipo del objeto se utiliza la función class()
- Clase: Define las estructura y comportamiento del objecto, ya sean vectores, matrices, listas, data frames, etc.
- Dimensión o longitud: Indica el número de elementos en un vector o lista, y en el caso de matrices o data frames las filas y columnas. Para obtener el número de elementos del objeto se utiliza las funciones length() y dim() respectivamente.

Los comandos en R, generalmente se componen de dos partes: objetos y funciones. Estos están separados por "<-". Por tanto, la forma general de un comando es:

Objeto<-función()

A continuación, vamos a ver las estructuras de datos más comunes en R y cómo crearlas, siendo estas vectores, matrices, listas y data.frames. También exploraremos como importar datos de diferentes fuentes externas.

### *2.5.1 Vector*

Un vector es la estructura de datos más sencilla en R. Existen tres tipos principales de vectores: numéricos, de caracteres y lógicos. Su característica principal es que todos los elementos tienen que ser del mismo tipo. Para crear un vector se utiliza la función c(), que permite concatenar elementos separados por comas.

```
# Creamos un vector con la función c()
> edad <- c(18, 23, 20, 17)  # Vector numérico
> edad
[1] 18 23 20 17
> deporte <- c(TRUE, FALSE, TRUE, TRUE)  # Vector lógico
> deporte
[1]  TRUE FALSE  TRUE  TRUE
> nombre <- c("Marcos", "Diego", "Manuel")  # Vector de caracteres
> nombre
[1] "Marcos" "Diego"  "Manuel"
```

Una vez creado el objeto podemos ver el tipo de dato con class(), acceder a los elementos del vector con objeto[n] y saber su longitud con length().

```
> class(deporte)  # Muestra el tipo de objeto
[1] "logical"
> edad[3]   # Accede al tercer elemento del vector edad
[1] 20
> length(edad) #Permite conocer la longitud del vector
[1] 4
```

También se pueden realizar operaciones aritméticas siempre que los vectores sean del mismo tipo y longitud.

```
> y <- edad / 2 # Divide cada elemento del vector edad entre 2
[1]  9.0 11.5 10.0  8.5
> v1=edad+y # Suma elemento a elemento los vectores edad e y
[1] 27.0 34.5 30.0 25.5
```

Otra forma de crear vectores es con secuencias, usando "from:to" o seq(from,to,by).

```
secuencia <- 1:10 # Crea una secuencia de 1 a 10
[1]  1  2  3  4  5  6  7  8  9 10
> secuencia1<-seq(1,10,2)  #Crea una secuencia de 1 a 10, donde va sum
ando de 2 en 2 hasta llegar a 10 o el valor más cercano.
```

```
[1] 1 3 5 7 9
```

### *2.5.2 Matriz*

Una matriz se puede describir como un vector multidimensional, compuesto por filas y columnas. Al igual que en el caso de los vectores, todos sus elementos deben ser del mismo tipo. Para crear la matriz se utiliza la funcion matrix(), la cual tiene los siguientes 4 argumentos principales:

1. data: Es el conjunto de valores que queremos dar a la matriz.
2. nrow: Especifica el número de filas de la matriz
3. ncol: Especifica el número de columnas de la matriz:
4. byrow: Indica el orden de cómo se organizan los datos en la matriz. Si ponemos TRUE, la matriz se llena de fila en fila (de izquierda a derecha). Si ponemos FALSE o no ponemos nada, se llena por columnas (de arriba hacia abajo)

Ejemplo de creación de matrices, ver atributos y acceder a los elementos de la matriz.

```
> # Matriz 3*2 rellenada por columnas, es decir, por defecto.
> ejmatriz_1 <- matrix(1:6,nrow=3 ,ncol=2 ,byrow=F)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> #Matriz 3*2 rellenada por filas, poniendo True en byrow.
> ejmatriz_2 <- matrix(1:6,nrow=3 ,ncol=2 ,byrow=T)
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
> class(ejmatriz_1) #Ver sus atributos
[1] "matrix" "array"
> dim(ejmatriz_1)  # Dimensión de la matriz
[1] 3 2
> ejmatriz_1[,2] # Extraer elementos de la matriz [fila, columna]
[1] 4 5 6
```

A continuación, se muestra como asignar nombres a las columnas y filas. También se puede realizar operaciones aritméticas como en los vectores, siempre que las matrices tengan las mismas dimensiones y sean del mismo tipo.

```
> #Asignar nombres a las columnas y filas respectivamente
> colnames(ejmatriz_1) <- c("Medalla oro","Medalla plata")
> rownames(ejmatriz_1) <- c("Manuel","María","Marco")
> ejmatriz_1
       Medalla oro Medalla plata
Manuel           1             4
Maria            2             5
Marco            3             6
> #función para trasponer una matriz con t()
> ejmatriz_3 <-t(ejmatriz_1)
              Manuel María Marco
```

```
Medalla oro         1     2     3
Medalla plata       4     5     6
```

### *2.5.3 Lista*

Según Rasilla (2023) las listas, al igual que los vectores, son estructuras de datos unidimensionales, sólo tienen largo, pero a diferencia de los vectores cada uno de sus elementos puede ser de diferente tipo o incluso de diferente clase, por lo que son estructuras heterogéneas. Podemos tener listas que contengan datos atómicos, vectores, matrices, arrays, data frames u otras listas.

Las listas se crean con objetos ya existentes con la función list(). A continuación, vamos a crear una lista que contiene un vector, matriz y un data frame que viene interno en R para posteriormente acceder a sus propiedades.

```
> # Primero creamos los objetos que queremos añadir en la lista
> numeros <- 10:30  # Vector
> tabla <- matrix(1:10, ncol = 5)  # Matriz
> datos <- PlantGrowth  # Data frame interno de RStudio
> # Creamos la lista dándole nombres a cada elemento
> lst <- list(vector= numeros, matriz = tabla, df = datos)
> # Propiedades de la lista
> class(lst)
[1] "list"
> length(lst)
[1] 3
```

Para ver los elementos que hay dentro de la lista utilizaremos [[]], y para acceder a un elemento dentro del vector, matriz o data frame utilizamos $, como se muestra a continuación.

```
# Acceder a los elementos de la lista con[[]]
> lst[[1]]  # Nos devuelve el primer elemento que es el vector
 [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
> lst[[2]]  # Aquí la matriz
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> #Para acceder a elementos más específicos utilizamos
> lst$vector[5]      # Acceder a un elemento dentro del vector
[1] 14
> lst$matriz[2,3]   # Acceder a un elemento de la matriz
[1] 6
> lst$df$weight[5] # Acceder a un elemento del data.frame
[1] 4.5
```

### *2.5.4 Data frame*

Un data frame es una estructura de datos muy utilizada por su similitud con las hojas de cálculo. Está compuesto por columnas y filas como una matriz, la diferencia es que cada

columna puede contener distintos tipos de datos (numéricos, de carácter y lógicos), por lo que lo convierte en una estructura heterogénea.

Para crear un data frame se utiliza la función data.frame(), donde cada columna es un vector con la misma longitud. A continuación, se muestran dos formas para crear data.frames.

```
> #Opcion 1 crear el data frame directamente
> liga_futbol <- data.frame(
+   nombre= c("Diego","Marcos","Manuel"),
+   edad=c(19,20,18),
+   goles= c(5,9,4),
+   balon_oro = c(5,9,4) > 7  )
> # Opción 2 agrupar los vectores previamente creados
> nombre= c("Diego","Marcos","Manuel")
> edad=c(19,20,18)
> goles= c(5,9,4)
> balon_oro = goles > 7
> liga_futbol2 <- data.frame(nombre,edad,goles,balon_oro)
# El resultado es el mismo sin importar la forma en la que se cree.
nombre edad goles balon_oro
1  Diego   19     5     FALSE
2 Marcos   20     9      TRUE
3 Manuel   18     4     FALSE
```

Una vez creado el data frame, podemos utilizar diferentes funciones como class(), dim(), length(), las cuales ya hemos visto en ejemplos con los vectores, matrices y listas. Con las siguientes funciones podemos obtener información relevante sobre el data frame.

```
names(liga_futbol)# Muestra el nombre de las columnas
[1] "nombre"    "edad"      "goles"     "balon_oro"
liga_futbol$nombre[2]#Extrae el elemento 2 del vector nombre
[1] "Marcos"
> str(liga_futbol)# Resumen de la estructura y tipos de datos del df
'data.frame':   3 obs. of  4 variables:
 $ nombre   : chr  "Diego" "Marcos" "Manuel"
 $ edad     : num  19 20 18
 $ goles    : num  5 9 4
 $ balon_oro: logi  FALSE TRUE FALSE
```

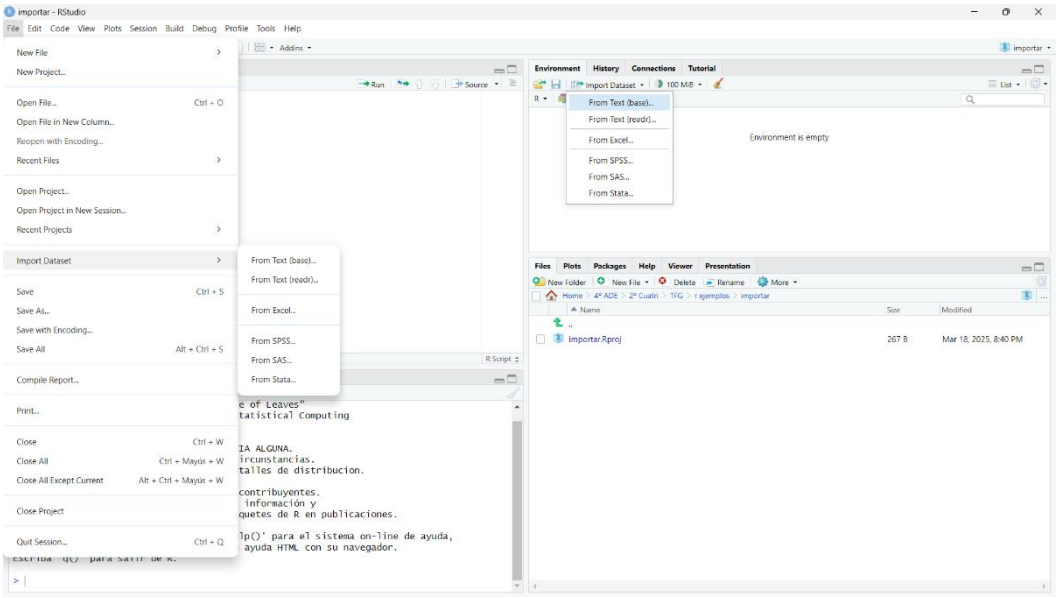
### *2.5.5 Importar datos.*

Anteriormente, hemos visto cómo introducir datos manualmente utilizando vectores, data frames, listas y matrices, lo cual es útil para entender cómo funcionan y para trabajar con pequeños conjuntos de datos. Sin embargo, si queremos hacer análisis econométricos con grandes volúmenes de datos y variables, la introducción manual resulta poco práctica y propensa a cometer errores. Por eso, en la práctica se suele trabajar con archivos de datos externos ya creados como Excel, CSV, SQL, ya que permiten un manejo de datos más eficiente, una transformación de datos más sencilla y por tanto reducen tanto las posibilidades de cometer errores como el tiempo invertido.

En R, tenemos dos opciones para importar datos:

- Utilizando código. Para importar datos, se suele utilizar las funciones read_excel(), read.csv(), read.table(), que forman parte del paquete readr (incluido en tidyverse). La elección de la función dependerá del tipo de archivo que queramos importar. En todo caso, deberemos especificar la ruta del archivo que desee cargar dentro de la función.
- Mediante desplegables. En Rstudio, podemos encontrar el desplegable para importar datos de dos maneras: desde la barra de tareas en File o directamente desde el panel de entorno, pinchando en Import Data. En ambas opciones, nos ofrece opciones de importar datos ya sean de texto, Excel, SPSS, SAS y Stata. La ventaja de este método es que permite ver los datos antes de importarlos, identificar errores y cambiar algunos aspectos como delimitadores, el rango etc. En la figura 4, vemos las dos opciones para importar datos mediante desplegables.

**Figura 4**
*Opciones para importar datos en RStudio*




***Nota.*** Elaboración propia

## 3 ANALISIS DESCRIPTIVO BASICO

En este apartado vamos a ver tanto las principales medidas de posición y dispersión (media, cuantiles, varianza, ...) como los principales gráficos en R (histograma, diagrama

de caja y diagrama de dispersión). Para ello vamos a utilizar la información del fichero Calif.xlsx, este fichero contiene una muestra de 420 distritos escolares de California. Estos datos incluyen variables como calificaciones medias, número de ordenadores, ratio de estudiantes por maestro, porcentaje de alumnos subvencionados, estudiantes que estudian inglés, entre otras. El objetivo de estos datos es analizar cómo factores económicos y educativos influyen sobre las calificaciones de los estudiantes.

## 3.1 Estadísticos

### *3.1.1 Medidas de Posición*

En la tabla 1 se observan los comandos que se utilizan tanto para las medidas de posición central, que indican los valores alrededor de los cuales se concentran la mayor parte de los datos en su conjunto (media, mediana y moda), como para las medias de posición no central, que se utilizan para dividir en partes iguales los conjuntos de datos (Cuantiles, máximo y mínimo)

**Tabla 1**

*Tabla de comandos en R de las medidas de posición principales*.

| Nombre del comando | Explicación |
|---|---|
| mean(data) | Media |
| median(data) | Mediana |
| mfv(data) | Moda |
| min(data) | Mínimo |
| max(data) | Máximo |
| quantile(data,0,25) | Primer cuartil |
| summary(data) | Min, Max, Mediana, Media, Q1 y Q3 |

***Nota***. Elaboración propia. Para calcular la moda necesitamos la librería "modeest"

A continuación, se proporciona un ejemplo con la variable computer, la cual muestra los ordenadores disponibles en cada distrito.

```
> mean(Calif$computer)
[1] 303.3833
> median(Calif$computer)
[1] 117.5
> mfv(Calif$computer)
[1] 50
> min(Calif$computer)
```

```
[1] 0
> max(Calif$computer)
[1] 3324
> quantile(Calif$computer,0.25)
25%
 46
> summary(Calif$computer)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    46.0   117.5   303.4   375.2  3324.0
```

### *3.1.2 Medidas de dispersión*

En la tabla 2 se presentan las funciones para calcular las medidas de dispersión principales (varianza, desviación típica y rango intercuartílico), que nos ayudan a saber si los valores del conjunto de datos están próximos entre sí o respecto a un valor central.

**Tabla 2**

*Tabla de comandos en R de las medidas de dispersión principales.*

| Nombre del comando | Explicación |
|---|---|
| var(data) | Varianza |
| sd(data) | Desviación típica |
| IQR(data) | Rango intercuartílico |

A continuación, se proporciona un ejemplo con la variable CalificacionExamen

```
> var(Calif$CalificacionExamen)
[1] 363.0301
> sd(Calif$CalificacionExamen)
[1] 19.05335
> IQR(Calif$CalificacionExamen)
[1] 26.6125
```

### *3.1.3 Medidas de dos variables*

En la tabla 3, se presenta el comando para calcular el coeficiente de correlación, el cual va a medir las posibles relaciones lineales entre dos variables cuantitativas.

**Tabla 3**

*Tabla de comandos para medidas de dos variables.*

| Nombre del comando | Explicación |
|---|---|
| cor(data) | Correlación |

Por ejemplo, para ver la correlación entre las variables computer y CalificacionExamen:

```
> cor(Calif$computer,Calif$CalificacionExamen)
[1] -0.07373576
```

### 3.2 Gráficos principales

Para generar gráficos elegantes y de alta calidad, vamos a utilizar ggplot2, que viene incluido dentro del paquete tidyverse. Este paquete se basa en la gramática de los gráficos, es decir, que se construye de capa en capa, lo que permite una personalización progresiva. Para crear un gráfico básico necesitamos 3 componentes esenciales:

- Datos: representan el conjunto de datos (data frame) que queremos representar.
- Aesthethic mapping (aes): en esta capa se establecen las variables que queremos representar en el gráfico, como por ejemplo los ejes, el color, forma y tamaño.
- Geometría (geom): nos indica el tipo de grafico a utilizar, ya sea histograma, diagrama de caja, diagrama de dispersión, etc.

La sintaxis obligatoria es:

```
library(ggplot2) # CARGAR EL PAQUETE

ggplot(data = nuestro_dataframe, aes(variable para el eje x, variable para el eje y)) + geom_funcion()
```

Además de estas capas se le pueden añadir capas adicionales, como facets, estadísticas , escalas, coordenadas y temas.

**Figura 5**
*Capas ggplot2*

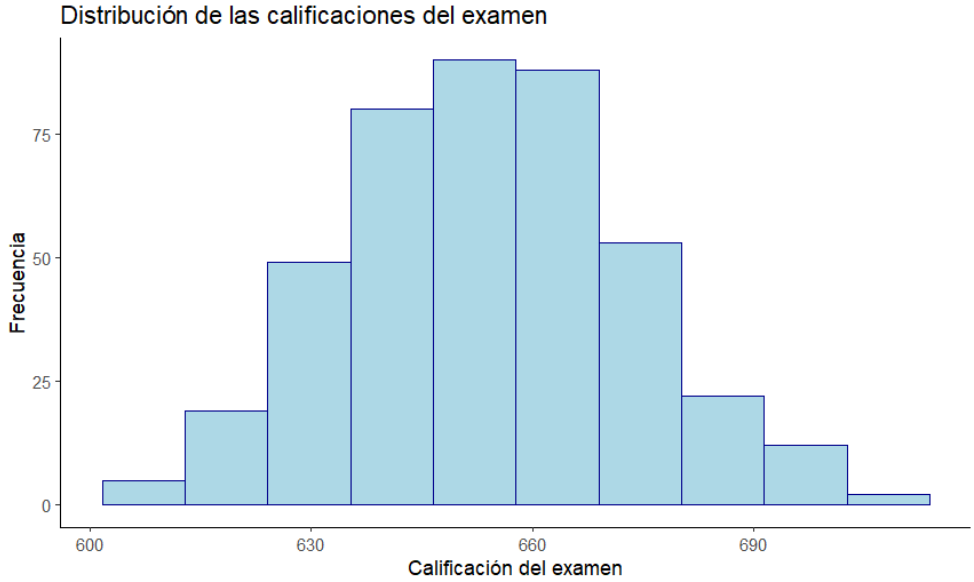

***Nota.*** Alonso, J. C. y Largo, M. F. (2022)

### *3.2.1 Histograma*

Un histograma es un gráfico que se utiliza para mostrar la distribución de una variable cuantitativa continua en formato de barras. En el eje horizontal se muestra los intervalos de valores y en el eje vertical la frecuencia absoluta. Este gráfico es útil para identificar tendencias, analizar simetría y dispersión de los datos. Para este gráfico utilizamos la función geom_histogram(). En este caso vamos a analizar la variable CalificaciónExamen.

**Figura 6**

*Histograma*




**Nota**. *Elaboración propia*

```
> ggplot(Calif, aes(CalificacionExamen)) +
   geom_histogram(color = "darkblue", fill = "lightblue", bins=10) +
   labs(
     title = "Distribución de las calificaciones del examen",
     x = "Calificación del examen",
     y = "Frecuencia"
   ) + theme_classic()
```

Como en todos los gráficos que vamos a ver a continuación, se utilizan las capas obligatorias: de datos, mapeo estético (aes) y la geometría (geom). Además, también incluiremos las funciones labs() para dar nombre a los títulos y ejes, así como theme_classic() para lograr un gráfico más limpio y visual.

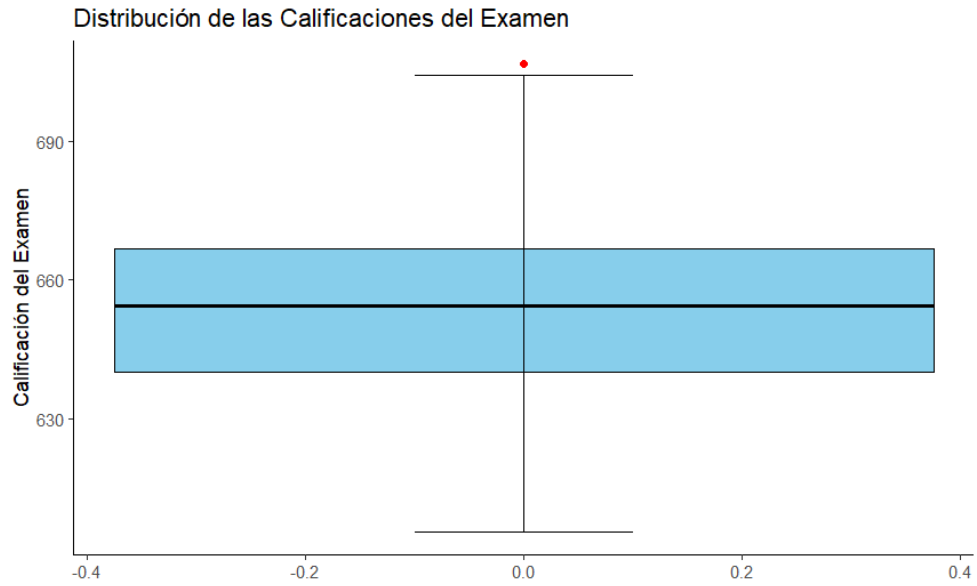
### *3.2.2 Diagrama de caja*

El diagrama de caja es una representación muy útil, ya que nos permite estudiar la posición, dispersión, simetría y analizar los valores atípicos.

Este grafico está basado en los cuartiles, es decir, la caja se extiende desde el primer hasta el tercer cuartil, ubicándose en su interior la mediana, que separa la caja en dos. Los bigotes son las líneas que van desde los extremos de la caja hasta los valores mínimo y máximo dentro de 1.5 veces el recorrido intercuartílico, mientras que los valores fuera de este rango se consideran atípicos. En este caso, analizaremos la variable CalificacionExamen con la función geom_boxplot().

**Figura 7**

*Diagrama de caja*




***Nota.*** Elaboración propia

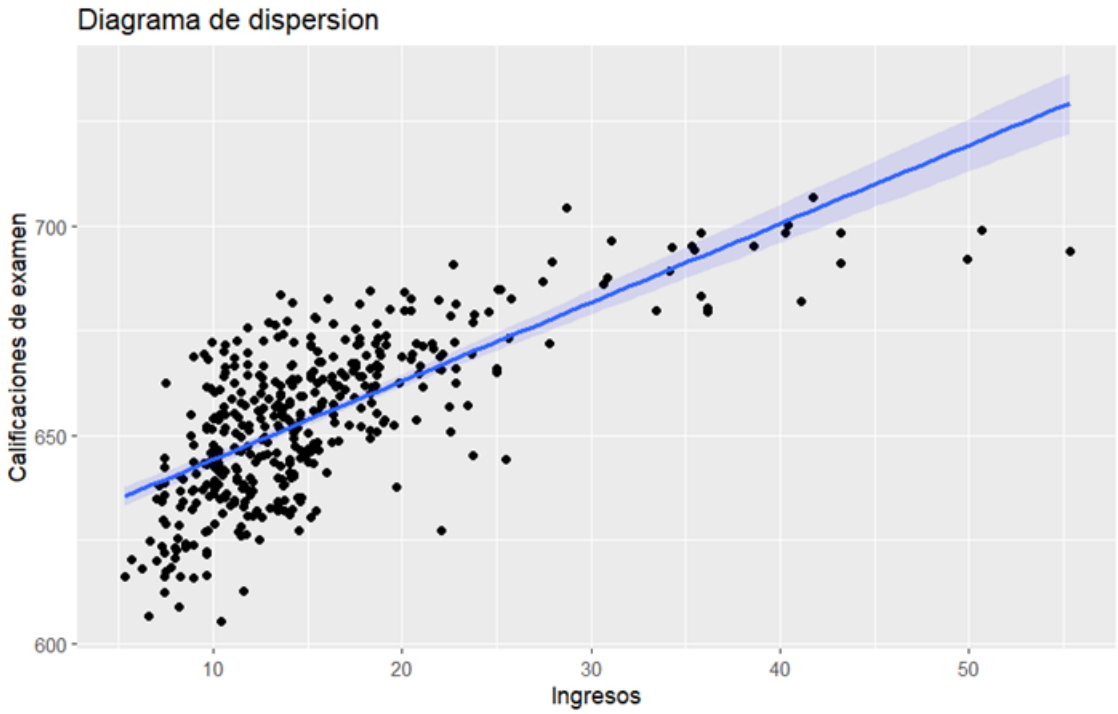
```
ggplot(Calif, aes(y = CalificacionExamen)) +
  geom_boxplot(fill= "skyblue", colour="black", outlier.colour="red")+
  stat_boxplot(geom = "errorbar", width = 0.2) +
  labs(title = "Distribución de las Calificaciones del Examen ",
       y = "Calificación del Examen") +
  theme_classic()
```

### *3.2.3 Diagrama de dispersión*

El diagrama de dispersión nos permite ver la relación entre variables numéricas. Cada observación se representa como un punto en el sistema de coordenadas cartesiano. La variable independiente se ubica en el eje horizontal y la dependiente en el vertical. Este gráfico nos permite identificar la relación entre las variables y su intensidad. En este caso, analizaremos la relación entre los ingresos y las calificaciones con la función geom_point() y también añadiremos una línea de tendencia con geom_smooth().

**Figura 8**

*Diagrama de dispersión*




**Nota.** En este grafico se puede observar que hay una relación positiva y lineal entre los ingresos y las calificaciones. Elaboración propia.

```
> ggplot(Calif, aes(avginc, CalificacionExamen)) +
    geom_point() +
    geom_smooth(method = "lm", alpha = 0.1, fill = "Blue") +
    labs(x = "Ingresos", y = "Calificaciones de examen", title = "Diagr
ama de dispersion" )
```

## 4 REGRESION MULTIPLE

En este apartado vamos a mostrar el análisis de regresión múltiple en el entorno de R aplicado al conjunto de datos de California que previamente hemos utilizado. El objetivo es ilustrar cómo estimar un modelo de regresión múltiple mediante mínimos cuadrados ordinarios, cómo crear variables ficticias, la realización de contrastes de hipótesis, la predicción y la comprobación de los supuestos clásicos (normalidad, homocedasticidad, multicolinealidad y linealidad).

### 4.1 Estimación

En esta sección vamos a tratar de explicar el comportamiento de las calificaciones (variable dependiente) utilizando un modelo de regresión múltiple.

Para estimar modelos lineales en R mediante MCO se utiliza la función lm(), siendo su sintaxis general la siguiente.

```
modelo <- lm (variable dependiente ~ variable independiente1 +
variable independiete2 + ..... n, data= nombre del data frame)
```

En nuestro caso, queremos explicar las calificaciones en función de los regresores PctEI (porcentaje de alumnos aprendiendo ingles), REM (ratio estudiante maestro) y PctCom (porcentaje de cumplimiento de requisitos para comedor subvencionado). Por lo tanto, el modelo teórico es:

$$Y = \beta_0 + \beta_1 REM + \beta_2 RctEI + \beta_3 PctCOM + \mu$$

```
modelo <- lm (CalificacionExamen ~ REM + PctEI + PctCom, data = Calif)
```

Una vez ejecutado, el modelo de regresión se guarda como un objeto, en este caso con el nombre modelo. Para obtener los resultados de la regresión introduciremos en la consola la función summary(objeto).

```
summary(modelo)
```

**Figura 9**

*Resultado regresión*

```
Call:
lm(formula = CalificacionExamen ~ REM + PctEI + PctCom, data = Calif)

Residuals:
    Min      1Q  Median      3Q     Max
-32.849  -5.151  -0.308   5.243  31.501

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 700.14997    4.68569 149.423  < 2e-16 ***
REM          -0.99831    0.23875  -4.181 3.54e-05 ***
PctEI        -0.12157    0.03232  -3.762 0.000193 ***
PctCom       -0.54735    0.02160 -25.341  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.08 on 416 degrees of freedom
Multiple R-squared:  0.7745,    Adjusted R-squared:  0.7729
F-statistic: 476.3 on 3 and 416 DF,  p-value: < 2.2e-16
```

**Nota**. Elaboración Propia.

En el resultado de la regresión (figura 9) se observa información de los residuos, los coeficientes de los estimadores junto a su error estándar, el valor del estadístico t y su respectivo p-valor, lo que permite analizar su significación individual. También, se muestra el $R^2$ y el $R^2$ ajustado, que indican la variabilidad explicada por el modelo. Por

último, se muestra el valor del estadístico F, el cual permite analizar la significación conjunta de los regresores.

En cuanto a las variables explicativas se observa que todas son significativas individualmente para un nivel de significación del 1%. El coeficiente de REM=-0,998 significa que, al aumentar en un alumno la ratio de estudiantes maestros, la calificación disminuye en 0,998 puntos, manteniéndose las demás variables constantes. Los demás coeficientes se interpretan de la misma forma, es decir, que un aumento de una unidad en PctEI y PctCOM van a tener un efecto negativo sobre las calificaciones. También se observa que el $R^2$ ajustado es de 0,7729, por lo que el 77,29% de la variabilidad total de la variable dependiente está explicada por el modelo. Por último, vemos que el p-valor del estadístico F es de 2,2e-16, indicando que las variables explicativas son conjuntamente significativas para explicar el modelo.

### 4.2 Variables ficticias

En los modelos econométricos, es habitual incluir variables cualitativas o categóricas (sexo, edad, categoría profesional…) que pueden ser relevantes para explicar la variable dependiente. Para añadir este tipo de información al modelo, se construyen de manera artificial variables que se denominan ficticas o dummy. Estas variables toman normalmente dos valores arbitrarios, 1 y 0, para indicar la presencia o no presencia de la característica, aunque se pueden tomar otros valores. En el caso de crear varias variables ficticias para utilizar diferentes modalidades de una variable categórica, hay que tener cuidado con la "trampa de las variables ficticias", por lo que habrá que incluir el número de modalidades menos una.

En nuestro caso vamos a utilizar la variable comp_stu (ordenadores por estudiante), y la vamos a transformar en una variable ficticia que tome el valor 1 cuando sea mayor a la media y 0 en el caso contrario. El objetivo es tratar de identificar si hay diferencias entre los centros con mayor y menor tecnología.

Para crear variables ficticias en R hay varios métodos. A continuación, mostraremos el más simple y directo. Primeramente, necesitamos crear el objeto de la media de los ordenadores por estudiante, después generamos una respuesta lógica (TRUE o FALSE)

indicando si es mayor o no a la media, y con la función as.numeric se transforma en valores binarios (TRUE=1 Y FALSE=0). Con la función table, como vemos a continuación, se aprecia que 170 observaciones superan la media y 250 están por debajo.

```
> mediacomp <- mean(Calif$comp_stu)
> comp_studummy<- as.numeric(Calif$comp_stu > mediacomp)
> table(comp_studummy)
comp_studummy
  0   1
250 170
```

Una vez creado el objeto de la variable ficticia, lo podemos añadir a la regresión como un regresor. Los resultados se muestran en la figura 10.

```
> modelofict <- lm (CalificacionExamen ~ REM + PctEI + PctCom + comp_s
tudummy, data = Calif)
```

**Figura 10**

*Implementación variable ficticia*

```
Call:
lm(formula = CalificacionExamen ~ REM + PctEI + PctCom + comp_studummy,
    data = Calif)

Residuals:
    Min      1Q  Median      3Q     Max
-33.364  -5.075  -0.110   5.574  30.672

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  697.54319    4.91842 141.823  < 2e-16 ***
REM           -0.90624    0.24425  -3.710 0.000235 ***
PctEI         -0.11584    0.03242  -3.573 0.000394 ***
PctCom        -0.54608    0.02156 -25.326  < 2e-16 ***
comp_studummy  1.60950    0.94357   1.706 0.088802 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.059 on 415 degrees of freedom
Multiple R-squared:  0.7761,    Adjusted R-squared:  0.7739
F-statistic: 359.6 on 4 and 415 DF,  p-value: < 2.2e-16
```

***Nota.*** Elaboración propia

Puesto que el p-valor asociado a la variable ficticia es 0,0888, no existen diferencias significativas al 5% en las calificaciones en función de si los centros tienen más o menos ordenadores por estudiante. Por lo tanto, en los siguientes apartados utilizaremos el modelo inicial sin la variable ficticia.

### 4.3 Contrastes de hipótesis

En este apartado nos vamos a centrar en los contrastes de hipótesis de la forma general, ya que hemos analizado anteriormente los contrastes de significación individual y conjunta en el summary del modelo, como se muestra en la figura 9.

A continuación, se describe cómo hacer todo tipos de contrastes utilizando la forma general del estadístico F, que se basa en comparar el modelo restringido frente al modelo original.

$$\frac{SSR_r - SSR_{ur}}{SSR_{ur}} * \frac{n-k-1}{q} \xrightarrow{H_0} F^H_{N-K-1}$$

Para aplicar este tipo de contraste en R necesitamos instalar y cargar el paquete car. En este ejemplo vamos a mostrar como comprobar si el efecto de REM es el mismo que el del PctEI, es decir:

$$\begin{cases} H_0: \beta_1 = \beta_2 \\ H_1: \beta_1 \neq \beta_2 \end{cases}$$

Se utiliza la función linearHypothesis() en la cual hay que poner el modelo estimado y después de la coma introducir la hipótesis nula.

**Figura 11**
*Resultado de contraste de hipótesis*

```
> linearHypothesis(modelo,"REM = PctEI")

Linear hypothesis test:
REM - PctEI = 0

Model 1: restricted model
Model 2: CalificacionExamen ~ REM + PctEI + PctCom

  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    417 35353
2    416 34298  1    1054.7 12.792 0.0003891 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

***Nota***. Elaboración propia.

Se puede ver que el p-valor es inferior a los niveles de significación habituales, por lo que se rechaza la hipótesis nula, es decir, los efectos de REM y PctEI sobre las calificaciones no son iguales.

### 4.4 Predicción

Una aplicación fundamental para los modelos de regresión es la predicción. Consiste en calcular el valor que tomaría la variable dependiente (CalificacionExamen) para determinados valores de los regresores (REM, PctEI, PctCom), usando la recta de regresión estimada.

Para predecir el valor de la variable dependiente del modelo que hemos estimado, debemos añadir una nueva observación con los valores de los regresores. Estos valores deben ir en formato data frame.

```
valoresdf<-data.frame(REM = 18, PctEI = 12, PctCom = 6 )
```

Una vez hemos dado valores a los regresores, utilizaremos la función predict() para sustituir los valores en el modelo estimado, y así obtener la predicción de la variable dependiente.

```
predict(modelo,valoresdf)
677.4374
```

Además, también podemos calcular tanto el intervalo de confianza como el intervalo de predicción con un nivel de significación del 95%, por defecto.

```
> predict(modelo,valoresdf,interval = "confidence" )
       fit     lwr      upr
1 677.4374 675.564 679.3108
> predict(modelo,valoresdf,interval = "prediction" )
       fit      lwr     upr
1 677.4374 659.4908 695.384
```

El intervalo de confianza nos da el rango donde se espera que se encuentre el promedio de las calificaciones para las variables explicativas dadas. En cambio, el intervalo de predicción nos da el rango en el que se espera que caiga el valor de una observación individual futura para las variables explicativas dadas.

### 4.5 Normalidad

Para validar el modelo, es fundamental comprobar si se cumple el supuesto de normalidad, es decir, que la perturbación del modelo se distribuye como una normal. Este supuesto garantiza que los estadísticos utilizados para los contrastes de significación individual y conjunta sigan distribuciones conocidas, lo cual permite que los contrastes

sean válidos. En caso de que no se cumpla, los contrastes no serán válidos para muestras pequeñas.

Para ver si se cumple la normalidad, se puede hacer mediante contrastes de hipótesis como los test de Jarque- Bera, Shapiro-Wilks y Kolmogorov Smirnov, al igual que mediante representaciones graficas como Q-Q. En los contrastes, la hipótesis nula indica que los residuos siguen una distribución normal y la hipótesis alternativa que los residuos no siguen una distribución normal.

A continuación, vamos a mostrar los resultados de los tres test. En el caso de Jarque - Bera, no viene implementado directamente en R, por lo que necesitaremos instalar y cargar el paquete tseries. Para la realización de los contrastes utilizaremos la función resid() que permite obtener los residuos del modelo estimado.

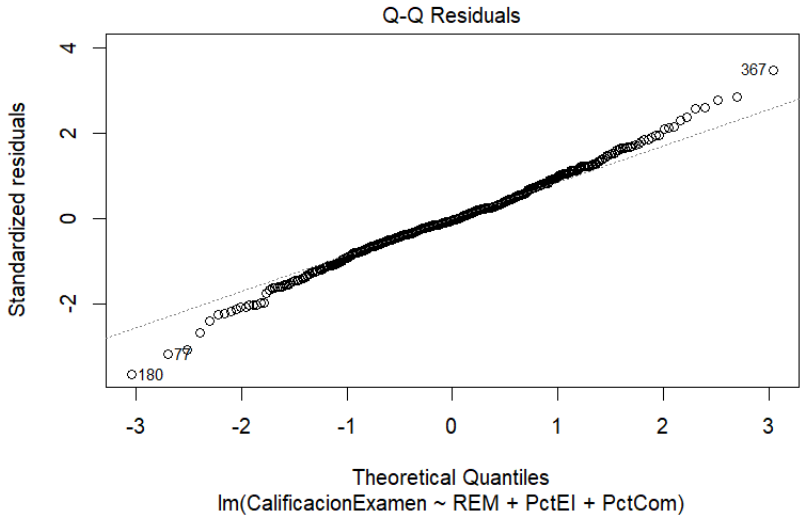
```
jarque.bera.test(resid(modelo))
X-squared = 10.626, df = 2, p-value = 0.004926
shapiro.test(resid(modelo))
W = 0.99279, p-value = 0.04097
ks.test(resid(modelo),"pnorm")
D = 0.37655, p-value < 2.2e-16
```

Como podemos observar, en todos los test llegamos a la misma conclusión: el p-valor es inferior a los niveles de significación habituales, por lo que se rechaza la hipótesis nula. Por lo tanto, los residuos no siguen una distribución normal.

Como hemos comentado antes, otra forma de ver si se cumple el supuesto es mediante la representación gráfica de cuantiles, comúnmente llamada grafico Q-Q. La función del grafico Q-Q es ver si un conjunto de datos sigue una distribución teórica especifica. En nuestro caso, utilizaremos como distribución teórica la normal. Su funcionamiento se basa en comparar los cuantiles de los residuos del modelo con los cuantiles de la distribución normal. Si los residuos se distribuyen como una normal, los puntos de la gráfica se deberían ajustar a la recta de 45 grados.

**Figura 12**

*Gráfica de cuantiles*




***Nota***. Elaboración propia

Como se observa llegamos a la misma conclusión que con los contrastes, ya que los residuos de cada observación no se ajustan a la recta completamente, ya que en los extremos se alejan. Esta forma podría darnos indicios de colas ligeramente pesadas, lo que podría deberse a valores atípicos o una mala especificación del modelo.

```
> plot(modelo,2)
```

Al utilizar la función plot() junto al objeto del modelo de regresión por MCO, se genera automáticamente cuatro gráficos útiles para analizar el cumplimiento de los supuestos. Si queremos ver de manera individual los gráficos se debe especificar su número correspondiente. En el trabajo hemos visto el grafico número 2, que permite evaluar el supuesto de normalidad. Posteriormente veremos el grafico 1, el cual es útil para evaluar si se cumplen los supuestos de homocedasticidad y linealidad.

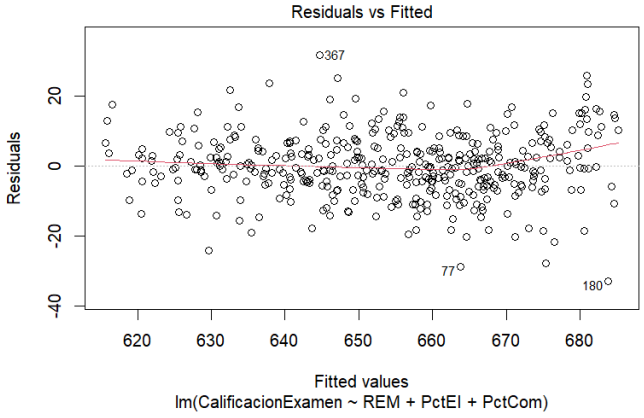
### 4.6 Heterocedasticidad

Otro supuesto clásico para validar el modelo es que la perturbación sea homoscedastica, esto significa que la varianza de las perturbaciones, condicionada a los valores de los regresores, es constante para todas las observaciones. Esto supone que la matriz de varianzas y covarianzas de la perturbación es $\sigma^2 I$ ,es decir, que en la diagonal las varianzas son constantes. Si este supuesto no se cumple, se presenta heterocedasticidad, es decir, la varianza condicionada de la perturbación no es constante. Las consecuencias de que haya heterocedasticidad es que el estimador no es óptimo y que no son válidos

los contrastes habituales. Para determinar si hay heterocedasticidad, existen dos métodos: el gráfico (del cual se puede sacar indicios) y las pruebas estadísticas.

Hay varias maneras gráficas de ver si hay indicios de heterocedasticidad. En nuestro caso vamos a verlo en un diagrama de dispersión donde en ordenadas se colocan los residuos de la estimación por MCO y en abscisas los valores ajustados.

**Figura 13**
*Gráfico heterocedasticidad*




**Nota**. Elaboración propia

Como se observa en el gráfico, la dispersión de la nube de puntos permanece constante, por lo que no hay indicios de heterocedasticidad.

```
plot(modelo,1)
```

Como hemos comentado anteriormente, de los gráficos se pueden sacar indicios, pero no conclusiones, por lo que es fundamental realizar pruebas estadísticas. El contraste que vamos a utilizar es el de White, ya que permite ver si la varianza de las perturbaciones condicionadas a los regresores es constante. Esta prueba implica el siguiente contraste de hipótesis:

$$\begin{cases} H_0: Homocedasticidad \\ H_1: Heterocedasticidad \end{cases}$$

El procedimiento del test de White es el siguiente:

1. Se estima el modelo original por MCO y se obtienen los residuos estimados.
2. Se realiza una regresión auxiliar donde la variable dependiente es el cuadrado de los residuos del modelo original y como variables explicativas se incluyen todas

las del modelo original, los cuadrados de dichas variables y sus productos cruzados.

3. Se calcula el siguiente estadístico que sigue una distribución asintótica $\chi^2$.

$$NR^2_{aux} \xrightarrow[N\to\infty]{} \chi^2_{k(aux)}$$

Para realizar el test de White en R necesitaremos instalar el paquete de lmtest, el cual proporciona la función bptest(). Esta función está hecha para realizar el contraste de Breuch- Pagan, pero si dentro añadimos los cuadrados de las variables explicativas y sus productos cruzados conseguimos aplicar el test de White, que es una prueba más general y capaz de detectar formas complejas de heterocedasticidad. En nuestro caso obtendremos 9 grados de libertad que equivalen al número de regresores del modelo auxiliar excluyendo el elemento constante.

```
> library(lmtest) # Cargar paquete

> bptest(modelo, ~ I(REM^2)+I(PctEI^2)+ I(PctCom^2)+REM*PctEI + REM*Pc
tCom + PctEI*PctCom, data=Calif)

        studentized Breusch-Pagan test

data:  modelo
BP = 30.26, df = 9, p-value = 0.0003962
```

Como podemos observar, a pesar de que el gráfico nos sugería que no había indicios de heterocedasticidad, el test de White nos indica que se rechaza la hipótesis nula, por lo que existe heterocedasticidad en el modelo.

Ante la presencia de heterocedasticidad en el modelo, vamos a estimar de forma robusta la matriz de varianzas y covarianzas del estimador MCO. Esto no va a eliminar la heterocedasticidad, pero permitirá corregir los errores estándar y solventar el problema de los contrastes, permitiendo que los estadísticos habituales t y F sean válidos asintóticamente, como muestra Heiss(2020).

Para estimar de forma robusta, debemos de tener instalados los paquetes car y lmtest. La función que utilizaremos es coeftest() que permite realizar pruebas estadísticas sobre los coeficientes del modelo de regresión. Dentro de la función, se especifica el modelo estimado y el argumento vcov=hccm(), que indica que se debe calcular de manera robusta la matriz de varianzas covarianzas en caso de heterocedasticidad. Por último,

hay que especificar el tipo de corrección robusta, en nuestro caso utilizaremos la hc0 que es la versión clásica del estimador robusto de White.

```
coeftest(modelo, vcov=hccm(modelo,type="hc0"))
t test of coefficients:

             Estimate Std. Error  t value  Pr(>|t|)
(Intercept) 700.149973   5.541874 126.3381 < 2.2e-16 ***
REM          -0.998310   0.268791  -3.7141 0.0002318 ***
PctEI        -0.121574   0.032675  -3.7207 0.0002260 ***
PctCom       -0.547345   0.023992 -22.8135 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

### 4.7 Multicolinealidad

Otro supuesto clásico del modelo de regresión lineal es comprobar que no exista correlación lineal entre las variables explicativas del modelo. Se pueden diferenciar dos tipos de multicolinealidad:

- Multicolinealidad perfecta, que se produce cuando uno de los regresores del modelo es una combinación lineal perfecta del resto de regresores del modelo. En este caso, se incumple el supuesto y es imposible calcular el estimador MCO.
- Multicolinealidad imperfecta, que surge cuando hay una fuerte correlación lineal entre los regresores. A diferencia del otro tipo de multicolinealidad, este no impide la estimación del modelo por MCO, pero si afecta a la precisión de las estimaciones.

Para detectar multicolinealidad existen varios métodos, en este caso nos enfocaremos en la matriz de correlación y el factor de inflación de varianza (VIF).

Para calcular la matriz de correlación del modelo, utilizaremos la función cor(). Dentro de la función especificamos el data frame y seleccionamos las variables del modelo mediante la utilización de un vector. Por último, indicamos el método Pearson, que mide la relación lineal entre las variables.

```
cor(Calif[,c("CalificacionExamen","REM", "PctEI", "PctCom")], method =
"pearson")
                  CalificacionExamen        REM      PctEI     PctCom
CalificacionExamen          1.0000000 -0.2263629 -0.6441239 -0.8687720
REM                        -0.2263629  1.0000000  0.1876425  0.1352035
PctEI                      -0.6441239  0.1876425  1.0000000  0.6530607
PctCom                     -0.8687720  0.1352035  0.6530607  1.0000000
```

Como podemos observar en la matriz de correlación no hay indicios de multicolinealidad ya que la correlación entre regresores no es significativamente alta. Además, los signos de la correlación entre los regresores y el regresando son negativos, al igual que los signos de los coeficientes del modelo estimado, reforzando los indicios de que no existe multicolinealidad.

La mejor forma para sacar conclusiones consistentes es utilizar el factor de inflación de la varianza (FIV), que recoge cuántas veces se ha incrementado la varianza de un coeficiente estimado por la existencia de un posible problema de multicolinealidad.

$$FIV = \frac{1}{1 - R_j^2}$$

Para calcular el FIV, únicamente utilizaremos la función vif() y dentro de ella seleccionaremos el nombre de nuestro modelo estimado.

```
> vif(modelo)
     REM    PctEI   PctCom
1.036795 1.774754 1.744149
```

Como podemos observar el FIV<10 en todas las variables explicativas, por lo que no existe multicolinealidad.

### 4.8 Linealidad

"Un modelo de regresión múltiple sufre de especificación incorrecta de la forma funcional cuando no explica de manera correcta la relación entre la variable dependiente y las variables explicativas observadas" (Wooldridge, 2010). Para detectar si el modelo está correctamente especificado, utilizaremos el contraste Reset Ramsey. Este contraste detecta si existe cualquier tipo de error de especificación, entre ellos, la linealidad.

Para este contraste se realiza una regresión auxiliar en la que se introduce al modelo original una matriz Z, que contiene las potencias de los valores ajustados $\hat{y}$. Se contrasta la hipótesis nula de linealidad frente a la hipótesis alternativa de no linealidad.

Para realizar el contraste de Reset en R, debemos cargar el paquete lmtest, el cual ofrece la función ressettest(). Dentro de esta función tenemos dos argumentos: power=2:3 que significa que añadiremos $\hat{Y}^2$ $y$ $\hat{Y}^3$ ; y type = fitted que significa que utilizaremos las potencias de los valores ajustados.

```
> resettest(modelo, power=2:3, type="fitted")

        RESET test

data:  modelo
RESET = 7.1749, df1 = 2, df2 = 414, p-value = 0.0008645
```

El contraste nos ha dado un p-valor menor a los niveles de significación habituales, rechazando la hipótesis nula. En consecuencia, se confirma que existe un problema de mala especificación en la forma funcional. Es decir, la especificación lineal no parece ser adecuada para explicar las calificaciones de los exámenes. Para solucionar esto, sería interesante incluir interacciones, variables polinómicas o realizar transformaciones con logaritmos.

## 5 CONCLUSIÓN

Este trabajo tiene como objetivo mostrar el funcionamiento del programa R para su aplicación en el análisis econométrico. En primer lugar, se mostró su interfaz y el manejo de estructuras de datos, para poder trabajar posteriormente de manera eficaz con un caso práctico. En el ejemplo utilizado, se construyó un modelo para explicar las calificaciones en California, lo que ha servido como guía para visualizar el proceso completo de un análisis econométrico. En este proceso se mostró cómo importar datos, así como estimar y validar el modelo mediante la utilización de diferentes paquetes y comandos.

A través del análisis de regresión múltiple en RStudio, se han podido comprobar sus múltiples funcionalidades y su eficacia para realizar análisis estadísticos y generar gráficos de alta calidad. Podemos señalar que una de sus principales ventajas es la amplia variedad de paquetes que ofrece, facilitando así la realización de diversas tareas estadísticas, la creación de gráficos personalizables y la manipulación de datos. Además, es un lenguaje de código abierto, gratuito y con una amplia comunidad, lo que hace que sea atractivo para una gran cantidad de usuarios. Sin embargo, encontramos algunas limitaciones, que hace que algunos usuarios escojan otros programas estadísticos como EViews o SPSS, ya que estos funcionan mediante menús desplegables y no necesitan escribir código para su funcionamiento.

Con todo lo expuesto, se puede concluir que R es una herramienta poderosa y versátil para realizar análisis estadísticos, y a pesar de que su curva de aprendizaje puede resultar

desafiante para perfiles sin conocimientos en programación, esto lo compensa con su gran ecosistema de paquetes que cubre la mayoría de los procedimientos estadísticos y su gran comunidad de usuarios, convirtiéndolo en uno de los softwares estadísticos del mercado más completos.

## 6 BIBLIOGRAFIA


Alonso, J. C. (2024). *Introducción al modelo clásico de regresión para científicos de datos en R*. Cali: Editorial Universidad Icesi. DOI: https://doi.org/10.18046/EUI/bda.h.4

Alonso, J. C. y Largo, M. F. (2022). *Empezando a visualizar datos con R y ggplot2*. Cali: Editorial Universidad Icesi. DOI: https://doi.org/10.18046/EUI/bda.h.3

Alonso, J. C. y Ocampo, M. P. (2022). *Empezando a usaR. Una guía paso a paso*. Cali: Editorial Universidad Icesi. DOI: https://doi.org/10.18046/EUI/bda.h.1

Daniel Paredes Inilupu. (2024c, diciembre 29). *Data Science con R*. https://bookdown.org/dparedesi/data-science-con-r/

Estructuras de datos en R. Universidad Carlos III de Madrid. https://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/ManualR/intro_estructurasdedatos.html#intro_estructuras

Field, A., Miles, J., & Field, Z. (2012). *Discovering statistics using R*. SAGE Publications.

Heiss, F. (2020). *Using R for Introductory Econometrics*.

Hothorn, T., & Everitt, B. S. (2009c). *A Handbook of Statistical Analyses Using R, Second Edition*. Chapman and Hall/CRC.

*How to Make Stunning Histograms in R: A Complete Guide with ggplot2*. (2024). Appsilon.com. https://www.appsilon.com/post/ggplot2-histograms

Material docente de Econometría I. Departamento de Economía Aplicada, Universidad de Valladolid.

Nahhas, R. W. (2025, 13 mayo). *2.3 Importing a file from another format | An Introduction to R for Research*. https://bookdown.org/rwnahhas/IntroToR/importing-a-file-from-another-format.html

Peng, R. D. (2022, 31 mayo). *R Programming for Data Science*. https://bookdown.org/rdpeng/rprogdatascience/

Rasilla, D. F. (2023, September 25). TEMA 1: ESTRUCTURAS DE DATOS EN r. https://personales.unican.es/rasillad/docencia/G14/TEMA_1/Estructuras_Datos.html

Rcoder. (2024, 25 marzo). *Box plot en R*. RCODER. https://r-coder.com/boxplot-en-r/

*RPubs - Objetos en R*. (s. f.-b). https://rpubs.com/Juli_Hdez36/940904

*RStudio User Guide - Get Started*. (2022, December 12). Docs.posit.co. https://docs.posit.co/ide/user/ide/get-started/

Stock, J. H., Watson, M. W., & Arrazola VacasMaría. (2012). *Introducción a la econometría (3a. ed.)*. Pearson Educación.

*The GNU Operating System and the Free Software Movement*. (s. f.). https://www.gnu.org/

Vega, J. B. M. (s. f.). *R para principiantes*. https://bookdown.org/jboscomendoza/r-principiantes4/

Verzani, J. (2014). *Using R for Introductory Statistics*. CRC Press.

Wooldridge, J. M. (2010). *Introducción a la econometría: un enfoque moderno*. Cengage Learning, Cop.