

Notas de Clase Estadística III 3009137

Capítulo 1: Breve introducción al R

Nelfi González Alvarez
Profesora Asociada

Departamento de Estadística
Universidad Nacional de Colombia, Sede Medellín



UNIVERSIDAD NACIONAL DE COLOMBIA

2025

Índice general

1. Breve introducción al R	1
1.1. Introducción	1
1.2. Cómo descargar e instalar R versión para Windows	1
1.3. Estructura del ambiente R	3
1.3.1. Ventana <i>R Console</i>	3
1.3.2. Ventana <i>Editor R</i>	4
1.3.3. Ventana <i>R Graphics: Device</i>	4
1.4. Organización y uso de funciones disponibles en R	4
1.5. Funciones R	7
1.5.1. Consulta de funciones disponibles en una librería	7
1.5.2. Consulta sobre uso de una función R (syntax)	7
1.5.3. Creación de funciones por el usuario	8
1.5.4. Listas de algunas funciones básicas de R	9
1.6. Objetos R	10
1.6.1. Asignación de valores y creación de objetos R	10
1.6.2. Algunos tipos de objetos que se pueden crear en R	11
1.6.3. Otros objetos R	11
1.7. Explicación de algunas funciones de interés	11
1.7.1. Para lectura de conjuntos de datos	11
1.7.2. Creando y extrayendo elementos de un vector en R	16
1.7.3. Creando y extrayendo elementos de una matriz o de un data.frame	17
1.7.4. Gráficas	18
Bibliografía	29

Índice de figuras

1.1.	Sitio de descarga del R	2
1.2.	Subdirectorios de descarga R para Windows	2
1.3.	Link última versión del R para Windows	2
1.4.	Instalador R	3
1.5.	ícono de inicio R	3
1.6.	Ventana <i>R Console</i>	3
1.7.	Ventana Editor R	4
1.8.	Ventana <i>R Graphics: Device</i>	5
1.9.	Ventana emergente usando comando <code>library()</code>	5
1.10.	Página de búsqueda y descarga de paquetes, según nombre	6
1.11.	Página de descarga paquete <code>car</code>	6
1.12.	Instalación paquete R usando archivo .zip	7
1.13.	Estructura de una función de usuario	8
1.14.	Vista de un Script R donde se editó líneas de programa creando un objeto escalar, un vector numérico, una matriz numérica y un marco de datos	11
1.15.	Vista de resultados en consola R de la ejecución de líneas de programa del Script R exhibido en Figura 1.14	12
1.16.	Vista ventanas Script R y Consola mostrando líneas de programa creando tres objetos R y el uso de la función <code>class()</code>	12
1.17.	Vista del contenido del archivo “average-weekly-male-earnings-in-editado.csv”	13
1.18.	Vista del contenido del archivo “average-weekly-male-earnings-in-editado.csv” pero abierto con bloc de notas	14
1.19.	Vista del contenido del archivo “ODONOVAN29.txt”	15
1.20.	Vista del contenido del archivo “cbe2.csv”	16
1.21.	Vista con resultados sobre comandos ejecutados sobre datos R <code>iris</code>	18
1.22.	Ilustración de la funciones <code>attach()</code> y <code>detach()</code> sobre datos R <code>iris</code>	18
1.23.	Ilustración de la función <code>plot</code> sobre variable cuantitativa vs. cuantitativa, sobre variable cualitativa y cuantitativa vs. cualitativa	20
1.24.	Ilustración de la función <code>plot</code> sobre el data.frame <code>iris</code>	20
1.25.	Ilustración de la función <code>boxplot()</code>	21
1.26.	Ilustración de la función <code>hist()</code>	22
1.27.	Ilustración de un histograma y box plot agregado en parte inferior, usando en la función <code>boxplot()</code> los argumentos <code>horizontal=TRUE</code> y <code>add=T</code>	22
1.28.	Ejemplos de uso de la función <code>layout()</code> para particionar una ventana gráfica. Las figuras (a) y (c) muestran cómo queda la partición o diseño de la ventana y las figuras (b) y (d) muestran cómo lucen las gráficas ubicadas en tales particiones, La partición realizada ha usado como argumento <code>mat</code> a <code>matriz=rbind(c(1,1,2,2),c(3,3,4,4))</code> . En las gráficas inferiores se ha usado además el argumento <code>heights=c(1,2)</code> , con el cual el alto de la fila inferior de la partición es 2 veces el alto de la fila superior.	24
1.29.	Ilustración de un gráfico de probabilidad normal, usando funciones <code>qqnorm()</code> , <code>qqline()</code>	25
1.30.	Ejemplos de uso de la función <code>decompose()</code> para filtrar componentes en una serie de tiempo. Las figuras (a) y (b) muestran las descomposiciones aditiva y multiplicativa, respectivamente; las figuras (c), (d) y (e), muestran por separado las componentes de tendencia, estacionalidad y residual, de la descomposición aditiva.	27

Capítulo 1

Breve introducción al R

1.1. Introducción

R es un lenguaje y ambiente que ofrece una amplia gama de métodos estadísticos y también puede ser considerado como un lenguaje de alto nivel. Entre otras cosas, permite definir funciones que pasan a ser parte del sistema, las cuales pueden ser usadas en sesiones posteriores. Entre sus ventajas, tenemos, su capacidad para operar con objetos, la programación en lenguaje matricial, la disponibilidad de una amplia base de operadores y la versatilidad que ofrece en la realización de gráficas.

El software está disponible bajo los términos de licencia GNU en forma de código fuente, para sistemas operativos Windows, Mac OS, UNIX y puede ser descargado en el website del R, donde se encuentran disponibles referencias importantes para aprender su uso, tales como,

1. *An Introduction to R: Notes on R: A Programming Environment for Data Analysis and Graphics, Version 4.4.2 (2024-10-31)*, by W. N. Venables, D. M. Smith and the R Core Team;
2. *R: A Language and Environment for Statistical Computing. Reference Index, Version 4.4.2 (2024-10-31)*, by The R Core Team.

1.2. Cómo descargar e instalar R versión para Windows

En la dirección <https://cran.r-project.org> ubicamos la página para descargar el R (ver Figura 1.1):

Download and Install R

Precompiled binary distributions of the base system and contributed packages, Windows and Mac users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for MacOS](#)
- [Download R for Windows](#)

Seleccione la opción [Download R for Windows](#). A continuación, visualiza la página **R for Windows**, donde tenemos las siguientes opciones (ver Figura 1.2):

Subdirectories:

- [base](#)
- [contrib](#)
- [old contrib](#)
- [Rtools](#)

Seleccione [base](#). En la página **R-4.4.2 for Windows** (ver Figura 1.3), seleccione [Download R-4.4.2 for Windows](#) (83 megabytes, 64 bit). Aparece luego la ventana de descarga. Guarde el archivo instalador.

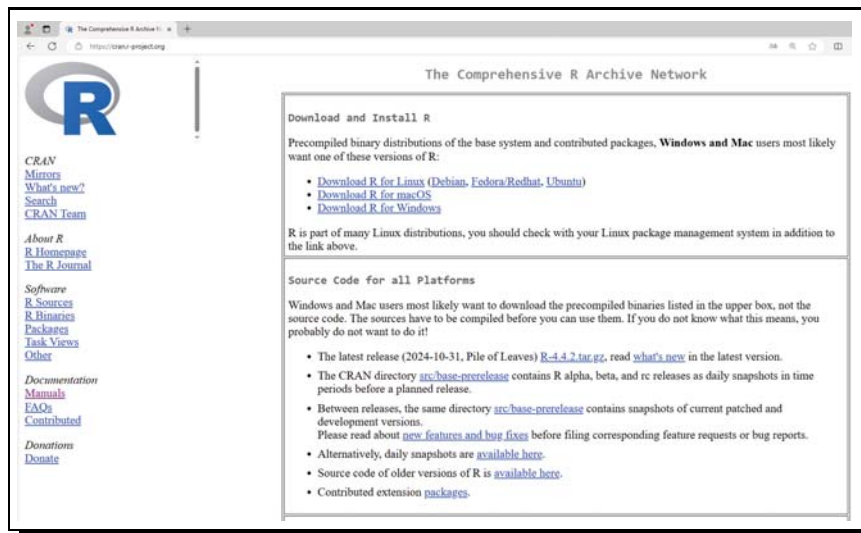


Figura 1.1: Sitio de descarga del R

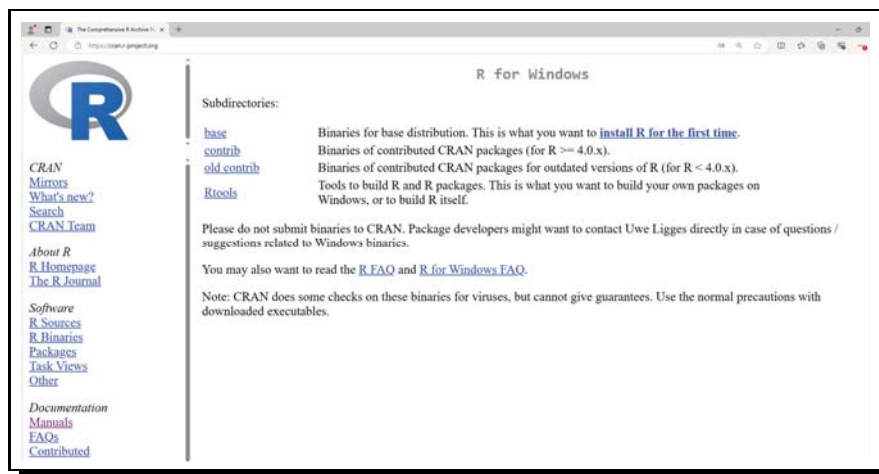


Figura 1.2: Subdirectorios de descarga R para Windows

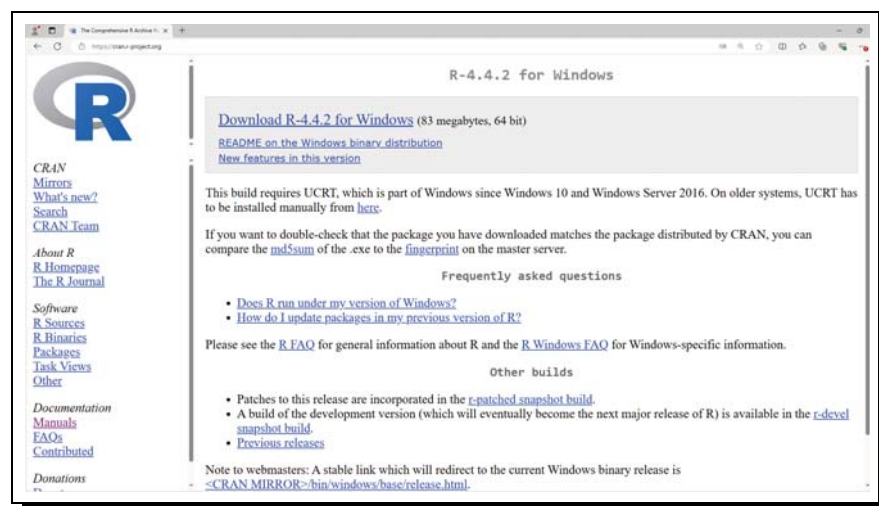


Figura 1.3: Link última versión del R para Windows

Nota 1.1. Para ejecutar el instalador en Windows 10, es mejor guardar el archivo instalador, para luego ejecutarlo como administrador, para ello, haga click derecho sobre ícono del archivo de instalación descargado y en la ventana emergente seleccione la opción “Ejecutar como administrador”.

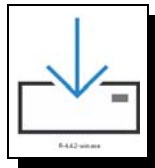


Figura 1.4: Instalador R

Nota 1.2. Siga paso a paso las instrucciones o preguntas del instalador. Cuando llegue a la ventana de selección de componentes a instalar, tenga cuidado de seleccionar primero los componentes referentes a manuales antes de proseguir.

1.3. Estructura del ambiente R

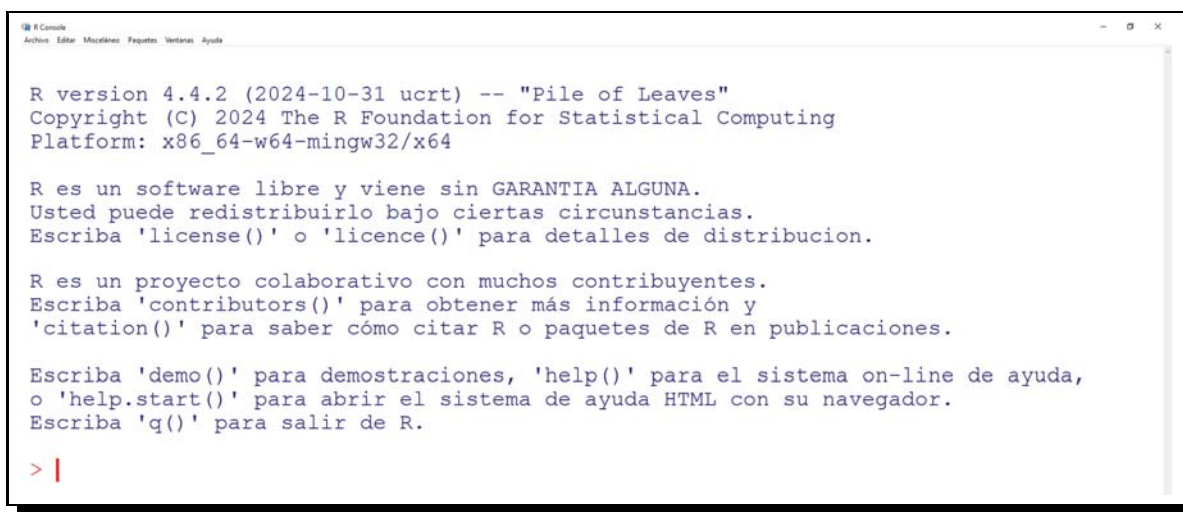
El ambiente R está estructurado por ventanas, a saber: Ventana *R-Console*, ventana *R Graphics: Device* y ventana *Editor R*.



Figura 1.5: ícono de inicio R

1.3.1. Ventana *R Console*

Al iniciar una sesión en R, el programa se abre en la ventana de comandos denominada *R Console*, ilustrada en la Figura 1.6, en la cual se pueden escribir líneas de comandos que serán ejecutados una vez se presione la tecla enter.

Figura 1.6: Ventana *R Console*

Nota 1.3. En Windows 10, es preferible iniciar como administrador, para ello en lugar de dar doble click sobre el ícono de inicio, haga click derecho sobre éste y en la ventana emergente seleccionar “Ejecutar como administrador”.

En la ventana *R Console* encontramos en la barra menú los siguientes menús:

- *Archivo*
- *Editar*

- *Misceláneo*
- *Paquetes*
- *Ventanas*
- *Ayuda*

R Console es una ventana de ejecución de comandos y de visualización de resultados numéricos, por eso, se prefiere editar primero cualquier programa en lenguaje R en un archivo de edición de R (llamado *Script*) antes de ejecutarlo. Los resultados visualizados en la consola se pueden copiar y pegar en cualquier archivo de edición como Word, con tipo de fuente Courier New para mejor presentación.

1.3.2. Ventana *Editor R*

Es una ventana de edición en lenguaje R. Se puede recurrir a la creación de un archivo de edición con el menú principal en la consola, *Archivo - Nuevo Script*, donde podemos editar nuestros programas específicos que deseemos y ejecutarlos en forma completa o parcialmente, usando las opciones disponibles en el menú *Editar* en esta ventana: *Correr línea o seleccionar*, *Ejecutar todo*, o bien, seleccionando las líneas a ejecutar y oprimir luego CTRL+R. En la Figura 1.7 se ilustra la ventana de edición con parte de un programa R escrito por el usuario. Estos archivos de edición pueden ser guardados con extensión *.R* y ser utilizados en cualquiera otra sesión cargándolos por el menú *Archivo - Abrir Script*.

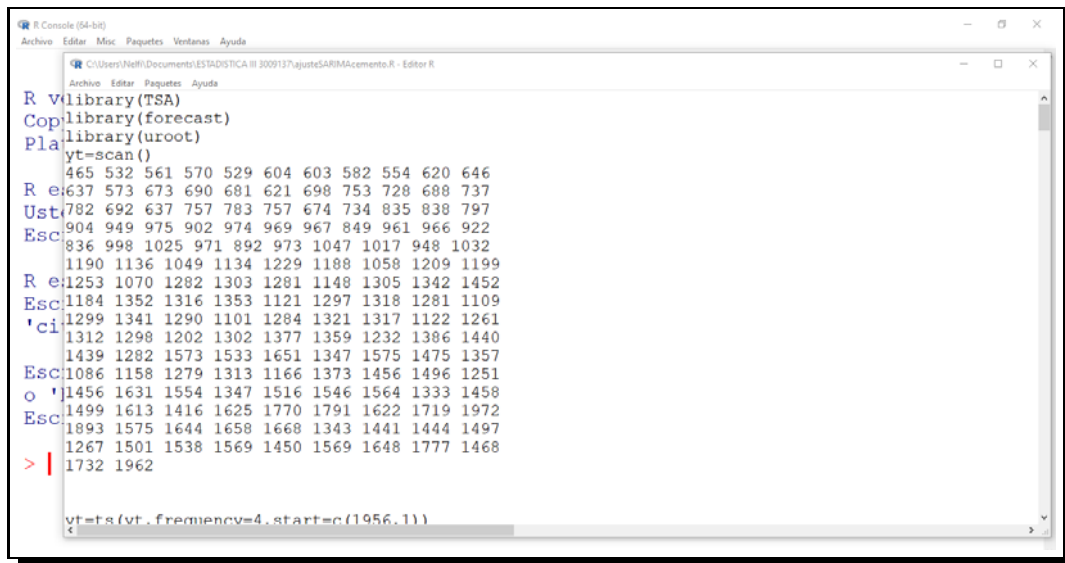


Figura 1.7: Ventana *Editor R*

1.3.3. Ventana *R Graphics: Device*

Se activa Cuando se utiliza una función que produce gráficos, por ejemplo, como se muestra en la Figura 1.8. Las gráficas pueden copiarse y guardarse en varios formatos gráficos, mediante el menú *Archivo* en esta ventana gráfica.

1.4. Organización y uso de funciones disponibles en R

En R las funciones están organizadas en librerías o paquetes. Por defecto, R inicializa en el paquete denominado *base*, en el cual encontramos las funciones básicas para el manejo de datos y gráficas. Existen otros paquetes en los cuales se encuentran herramientas de análisis más especializadas, las cuales pueden ser utilizadas cargando previamente la librería que las contiene. Una librería o paquete puede cargarse mediante la función `library()` o bien con `require()` o a través del menú *Paquetes - Cargar paquetes...*

Una lista de las librerías disponibles en su máquina puede obtenerse ejecutando `library()`, lo que permite ver, en una ventana denominada *R packages available*, un listado como ilustra la Figura 1.9.

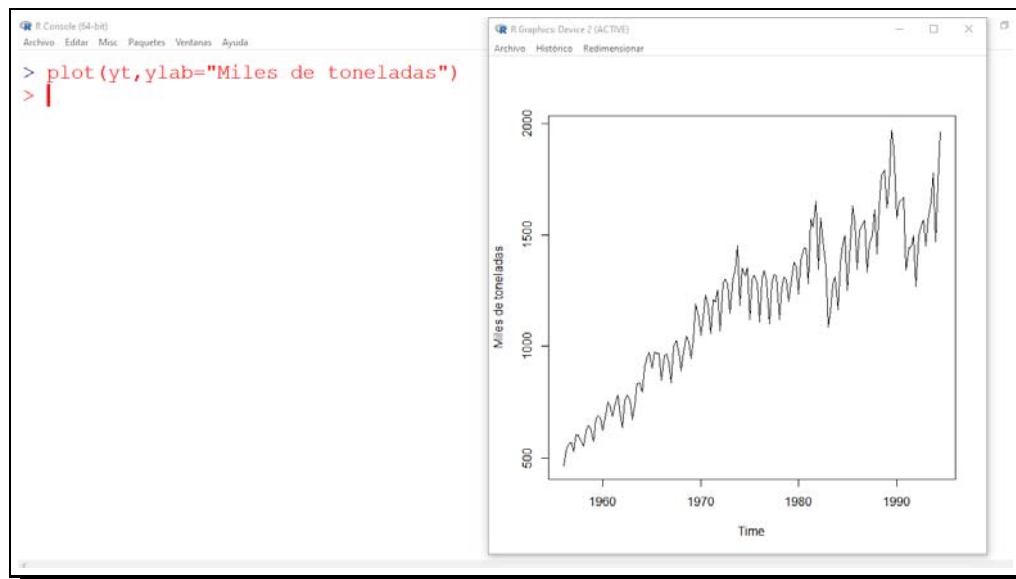


Figura 1.8: Ventana R Graphics: Device

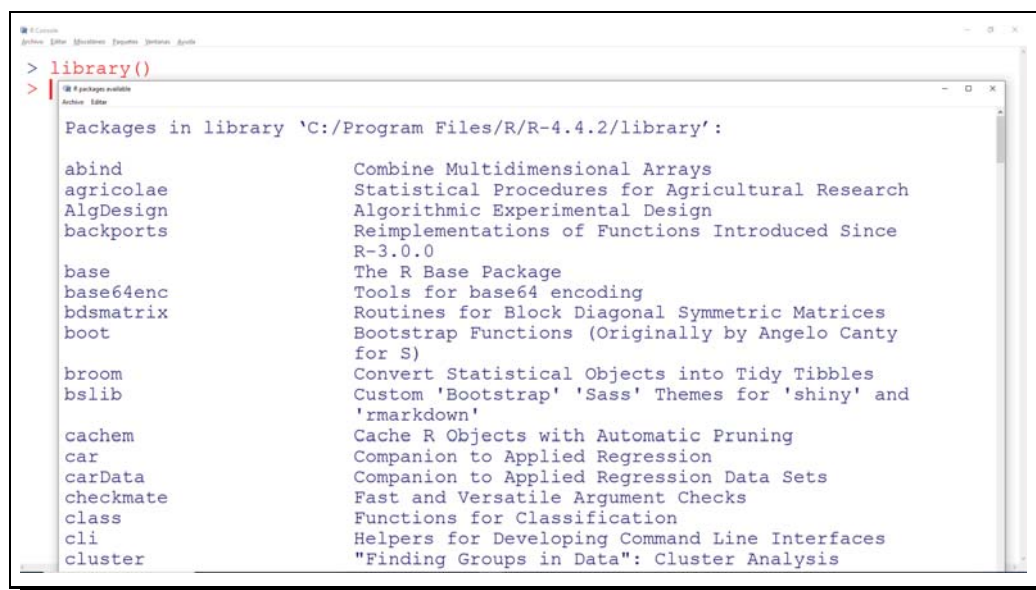


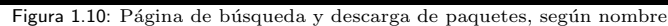
Figura 1.9: Ventana emergente usando comando library()

Si alguna librería es invocada sin antes descargarla del sitio Web de R, se producirá un mensaje de error. Existe una gran cantidad de paquetes disponibles en el website del R que el usuario puede descargar simplemente siguiendo los siguientes pasos.

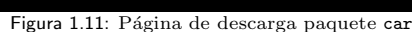
1. Conéctese a internet.
2. Inicialice una sesión R. Tenga en cuenta que en windows 10 o posterior, para descargar librerías durante una sesión, se recomienda iniciar la sesión R ejecutando el programa como administrador, de lo contrario, los archivos asociados a las librerías R no tendrán permiso para instalarse en la ruta apropiada.
3. Vaya al menú *Instalar paquete(s)...* En la ventana emergente *CRAN mirror*, seleccione *Colombia (Cali)* [https] y presione OK.
4. En la ventana *Packages* resultante, seleccione el paquete que se desea descargar, buscando por nombre en orden alfabético y finalmente presione OK.

Sin embargo, lo anterior a veces puede fallar, bien sea por error de configuración del R o problemas con la conexión a Internet usando R. En este caso, puede realizarse lo siguiente:

- Table of available packages, sorted by date of publication
- Table of available packages, sorted by name



4. Una vez seleccionado el paquete en la lista alfabética, aparece la página de descarga, por ejemplo, para la librería **car**, la página aparece titulada como **car: Companion to Applied Regression**, ilustrada en la Figura 1.11, donde se visualiza una breve descripción de la librería, su número de versión y dependencia con otras librerías o paquetes que deberán descargarse, caso que el usuario aún no las tenga disponibles en su máquina. En la sección **Downloads** se selecciona la opción para *Windows binary: r-release: car.3.1-3.zip* (este archivo cambiará de nombre de acuerdo a sus actualizaciones). Guarde el archivo en su máquina.



- Para instalar en R la librería descargada mediante archivo .zip, inicie una sesión R (recuerde en Windows 10 iniciar R como administrador), vaya al menú *Paquetes - Instalar paquete(s) a partir de archivos zip locales...* Emerge una ventana como la ilustrada en la Figura 1.12 para que ud. explore en su máquina hasta hallar la ruta donde guardó el archivo .zip del paquete R a instalar.

1.5. Funciones R

Una función R se reconoce por una cadena de caracteres que corresponde al nombre de la función, seguida de paréntesis () dentro de los cuáles se especifica el objeto(s) R sobre el que se aplica la función y una serie de argumentos que permiten activar o modificar la forma en que la función opera y presenta sus resultados. Por ejemplo, `plot(x,y)` produce un gráfico de dispersión entre los objetos R denominados `x`, `y`, que se supone fueron creados previamente, colocando en el eje horizontal al objeto `x` y en el vertical al objeto `y`.

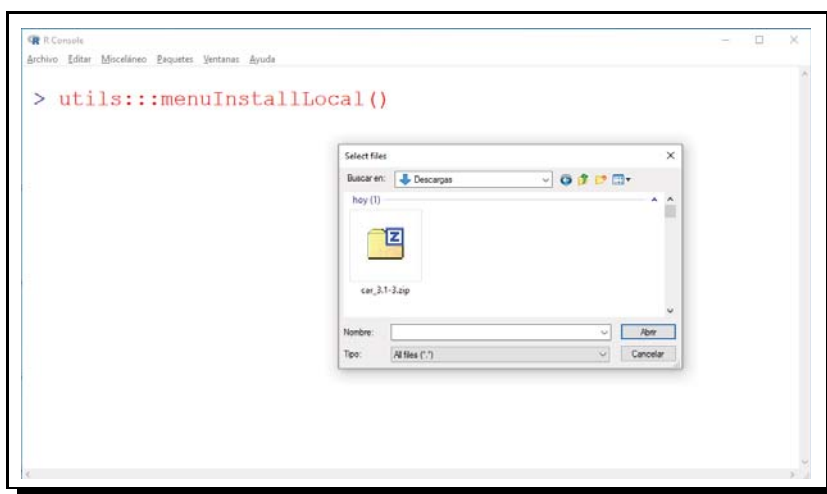


Figura 1.12: Instalación paquete R usando archivo .zip

1.5.1. Consulta de funciones disponibles en una librería

Para consultar la lista de funciones disponibles en una librería particular (ya descargada e instalada) usamos el comando:

```
library(help="nombre de la libreria")
```

Por ejemplo, si se desea saber cuáles funciones están disponibles en la librería `splines`, ejecutamos el siguiente comando:

```
library(help="splines")
```

1.5.2. Consulta sobre uso de una función R (sintaxis)

Para conocer sobre la sintaxis y uso de alguna función en particular, podemos usar el menú *Ayuda* o el comando `?paquete::función`, por ejemplo,

```
?car::boxCox.
```

Si la función se encuentra en la librería `base`, usamos simplemente el comando `?función`, por ejemplo,

```
?lm
```

Nota 1.4. R es sensible a mayúsculas y minúsculas, por lo que es importante escribir los nombres de funciones y librerías conforme han sido definidos.

Algunas veces, el usuario desconoce el nombre de la función deseada. Si ya existe en alguna librería previamente descargada, se puede recurrir a la búsqueda con una palabra clave, con `help.search("palabra-clave")` ó `??palabra-clave`.

Por ejemplo,

```
??shapiro ó help.search("shapiro"),
```

da como resultado un listado de funciones R y su ubicación (`paquete::función`), como el siguiente,

Help files with alias or concept or title matching 'shapiro' using fuzzy matching:

```
nortest::sf.test           Shapiro-Francia test for normality
stats::shapiro.test       Shapiro-Wilk Normality Test
Aliases: shapiro.test
```

Type 'PKG::FOO' to inspect entries 'PKG::FOO', or 'TYPE?PKG::FOO' for entries like 'PKG::FOO-TYPE'.

1.5.3. Creación de funciones por el usuario

R permite al usuario crear sus propias funciones con el comando `function()`, éstas son almacenadas en una forma interna especial y pueden ser usadas en expresiones subsiguientes. Esta característica proporciona una gran potencialidad y versatilidad al lenguaje permitiendo al usuario mayor productividad en el uso del R. Una función es definida mediante una asignación de la forma indicada en la siguiente figura.

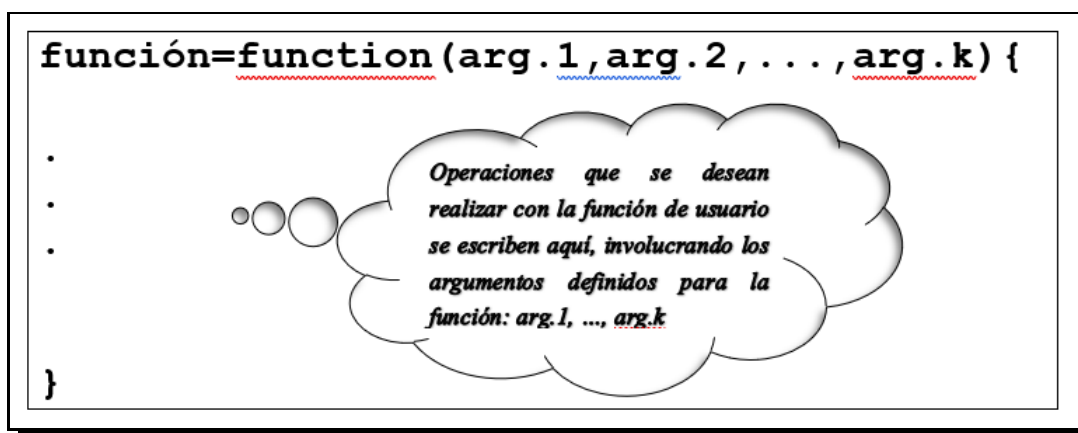


Figura 1.13: Estructura de una función de usuario

Las expresiones usadas en las operaciones internas, aplicadas sobre los argumentos `arg.i`, para obtener algún valor o producir un objeto determinado (vector, matriz, lista, arreglo, etc.), deben ser admisibles en R. Una invocación o llamada de la función toma la forma `nombre-función(expr.1, expr.2, ...)`, donde `expr.i` es el valor particular para el argumento `arg.i` de la función. Por ejemplo, como se ilustra el siguiente bloque de programación, se crean la función de usuario: `amplitud.cobertura` para el cálculo de la amplitud promedio y la proporción de cobertura, de intervalos de pronóstico generados con algún modelo previamente ajustado, donde los argumentos `LIP` y `LSP` corresponden a vectores con los límites inferior y superior de un conjunto de pronósticos por intervalos, en tanto que el argumento `real`, se refiere al vector con los correspondientes valores reales observados.

```
#Creación de la función
amplitud.cobertura=function(real,LIP,LSP){
a=LSP-LIP
am=mean(a)
I=ifelse(real>=LIP & real<=LSP,1,0)
p=mean(I)*100
res=list(Amplitud=am,"Cobertura(%)"=p)
unlist(res)
}
```



```
#Aplicación sobre pronósticos previamente calculados
amplitud.cobertura(real=ytf,LIP=predmod[,2],LSP=predmod[,3])
```

En el anterior ejemplo, el objeto `predmod` supone un objeto matriz de predicciones con tres columnas: predicción puntual, límite inferior y límite superior de pronóstico; el objeto `ytf` supone un vector con los valores reales para los cuales se calculó el pronóstico.

En la siguiente Sección, se presenta un listado de las funciones básicas utilizadas o que pueden ser útiles para el lector.

1.5.4. Listas de algunas funciones básicas de R

Funciones para entrada y lectura de datos y escritura y exportación de resultados y objetos R

Comando	Función
<code>scan()</code>	Lectura de datos. Especial para datos no estructurados
<code>read.table()</code>	Lectura de archivos en formato de tabla
<code>read.fwf()</code>	Lectura de archivos en formato de tabla con ancho fijo
<code>read.csv()</code>	Lectura de archivos en formato de tabla con datos separados por comas
<code>read.ftable()</code>	Lectura de una tabla de contingencia guardada en archivo ASCII
<code>sink()</code>	Envío a archivo ASCII los resultados de una sesión
<code>write()</code>	Escribe una matriz en un archivo ASCII
<code>write.ftable()</code>	Escribe una tabla de contingencia en un archivo ASCII
<code>xtable()</code>	Escribe una matriz en formato LaTeX. Requiere la librería <code>xtable</code>
<code>fable()</code>	Permite presentar decentemente un arreglo multidimensional

Operadores

Aritméticos	De comparación	Lógicos y de control
<code>+</code> Suma	<code><</code> menor	<code>&</code> y
<code>-</code> Resta	<code>></code> mayor	<code> </code> ó
<code>*</code> Multiplicación	<code><=</code> menor o igual	<code>!</code> no
<code>/</code> División	<code>>=</code> mayor o igual	<code>all(...)</code> ¿Todos los valores lógicos son ciertos?
<code>^</code> Exponenciación	<code>==</code> igual	<code>any(...)</code> ¿Alguno de los valores lógicos es cierto?
<code>%/%</code> División entera	<code>!=</code> diferente	<code>&&</code> Si primer operando es cierto evalúa segundo operando
<code>%%</code> Operador módulo		<code> </code> Si primer operando es falso evalúa segundo operando.
<code>sqrt(x)</code> \sqrt{x}		
<code>exp(x)</code> e^x		

Funciones relacionadas con distribuciones

Distribución	Densidad	Función Acumulada	Cuantil p	Números aleatorios
Uniforme	<code>dunif(x,...)</code>	<code>punif(q,...)</code>	<code>qunif(p,...)</code>	<code>runif(n,...)</code>
Normal	<code>dnorm(x,...)</code>	<code>pnorm(q,...)</code>	<code>qnorm(p,...)</code>	<code>rnorm(n,...)</code>
Binomial	<code>dbinom(x,...)</code>	<code>binom(q,...)</code>	<code>qbinom(p,...)</code>	<code>rbinom(n,...)</code>
Lognormal	<code>dlnorm(x,...)</code>	<code>plnorm(q,...)</code>	<code>qlnorm(p,...)</code>	<code>rlnorm(n,...)</code>
Beta	<code>dbeta(x,...)</code>	<code>pbeta(q,...)</code>	<code>qbeta(p,...)</code>	<code>rbeta(n,...)</code>
Geométrica	<code>dgeom(x,...)</code>	<code>pgeom(q,...)</code>	<code>qgeom(p,...)</code>	<code>rgeom(n,...)</code>
Gamma	<code>dgamma(x,...)</code>	<code>pgamma(q,...)</code>	<code>qgamma(p,...)</code>	<code>rgamma(n,...)</code>
Ji cuadrado	<code>dchisq(x,...)</code>	<code>pchisq(q,...)</code>	<code>qchisq(p,...)</code>	<code>rchisq(n,...)</code>
Exponencial	<code>dexp(x,...)</code>	<code>pexp(q,...)</code>	<code>qexp(p,...)</code>	<code>rexp(n,...)</code>
F	<code>df(x,...)</code>	<code>pf(q,...)</code>	<code>qf(p,...)</code>	<code>rf(n,...)</code>
Hipergeom.	<code>dhyper(x,...)</code>	<code>phyper(q,...)</code>	<code>qhyper(p,...)</code>	<code>rhyper(n,...)</code>
t	<code>dt(x,...)</code>	<code>pt(q,...)</code>	<code>qt(p,...)</code>	<code>rt(n,...)</code>
Poisson	<code>dpois(x,...)</code>	<code>ppois(q,...)</code>	<code>qpois(p,...)</code>	<code>rpois(n,...)</code>
Weibull	<code>dweibull(x,...)</code>	<code>pweibull(q,...)</code>	<code>qweibull(p,...)</code>	<code>rweibull(n,...)</code>
Binom. Neg.	<code>dnbinom(x,...)</code>	<code>pnbinom(q, s,...)</code>	<code>qnbinom(p,...)</code>	<code>rnbinom(n,...)</code>

Funciones que producen escalares

Comando	Función
<code>max()</code>	Máximo del argumento
<code>min()</code>	Mínimo del argumento
<code>sum()</code>	Suma de todos los elementos del argumento
<code>mean()</code>	Promedio aritmético de todos los elementos del argumento
<code>var()</code>	Varianza de todos los elementos del argumento, cuando éste es un vector, o matriz de covarianzas si el argumento es una matriz
<code>median()</code>	Mediana del argumento
<code>quantile(...,probs=c(...))</code>	Cuantiles del argumento con las proporciones indicadas en 'probs'
<code>prod()</code>	Producto de todos los elementos del argumento
<code>length()</code>	Número de elementos del argumento si este es una lista o vector
<code>ncol()</code>	Número de columnas si el argumento es una matriz
<code>nrow()</code>	Número de filas si el argumento es una matriz

Condicionales y loops

- `if(condición){expresiones} else expresión`

- `ifelse(condición,1,0)`
- `for(nombre in expresión){expresiones}`
- `while(condición){expresiones}`

Para creación de algunos objetos R

Vectores, matrices y arreglos	
Comando	Función
<code>c()</code>	Crea vectores
<code>append()</code>	Combina vectores o adiciona elementos a un vector
<code>matrix(), as.matrix(), data.matrix()</code>	Crea matrices
<code>array(), as.array()</code>	Crea arreglos
Listas y tramas de datos	
Comando	Función
<code>list(), as.list()</code>	Crea listas de objetos
<code>data.frame(), as.data.frame()</code>	Crea colecciones de variables en estructura tabular

Algunas operaciones con matrices

Comando	Función
<code>/%*%</code>	Producto matricial
<code>t()</code>	Transposición de una matriz
<code>crossprod(A)</code>	Producto $A^t A$
<code>svd()</code>	Descomposición en valores singulares de una matriz
<code>qr()</code>	Descomposición qr
<code>chol()</code>	Descomposición de Cholesky
<code>solve()</code>	Inversa de una matriz
<code>cbind()</code>	Combina matrices por columnas
<code>rbind()</code>	Combina matrices por filas
<code>eigen()</code>	Cálculo de valores y vectores propios
<code>diag()</code>	Crea una matriz diagonal si el argumento es un vector o retorna la matriz diagonal de una matriz

Aplicando funciones a objetos R

Comando	Función
<code>apply()</code>	Aplica una función a filas o columnas de una matriz
<code>tapply()</code>	Aplica una función a cada celda de un arreglo
<code>lapply()</code>	Aplica una función a cada elemento de una lista y devuelve una lista
<code>sapply()</code>	Como lapply pero devuelve un vector o matriz

Funciones que producen gráficas

Comando	Función
<code>hist()</code>	Grafica histogramas
<code>boxplot()</code>	Grafica boxplots
<code>plot()</code>	Función genérica para gráficos de dispersión, de series de tiempo, de residuales, etc.
<code>qqplot()</code>	Gráfico cuantil-cuantil
<code>qqnorm()</code>	Gráfico de probabilidad normal
<code>pairs()</code>	Gráfico de matrices de dispersión

Funciones usadas en regresión lineal

Comando	Función
<code>lm()</code>	Función para ajuste de un modelo lineal por mínimos cuadrados
<code>summary()</code>	Función genérica para exhibir resumen de resultados de modelos ajustados
<code>residuals()</code>	Extrae residuales de modelos ajustados con alguna función como <code>lm()</code>
<code>fitted()</code>	Extrae valores ajustados de modelos ajustados con alguna función de modelación
<code>predict()</code>	Produce predicciones a partir de los resultados de varias funciones R que ajustan modelos. los métodos que usa dependen de la clase de objeto R producido por una función de modelación específica.
<code>durbinWatsonTest()</code>	Función de la librería <code>car</code> que realiza el test Durbin-Watson

Nota 1.5. Existen muchas otras funciones para el análisis y ajuste de modelos de regresión que serán vistas posteriormente en el desarrollo del curso. Recuerde que puede consultar la sintaxis de cualquier función R siguiendo lo explicado en la Sección 1.5.2.

1.6. Objetos R

1.6.1. Asignación de valores y creación de objetos R

El símbolo de asignación es el igual, `=`, pero también podemos usar `<-`, por ejemplo,

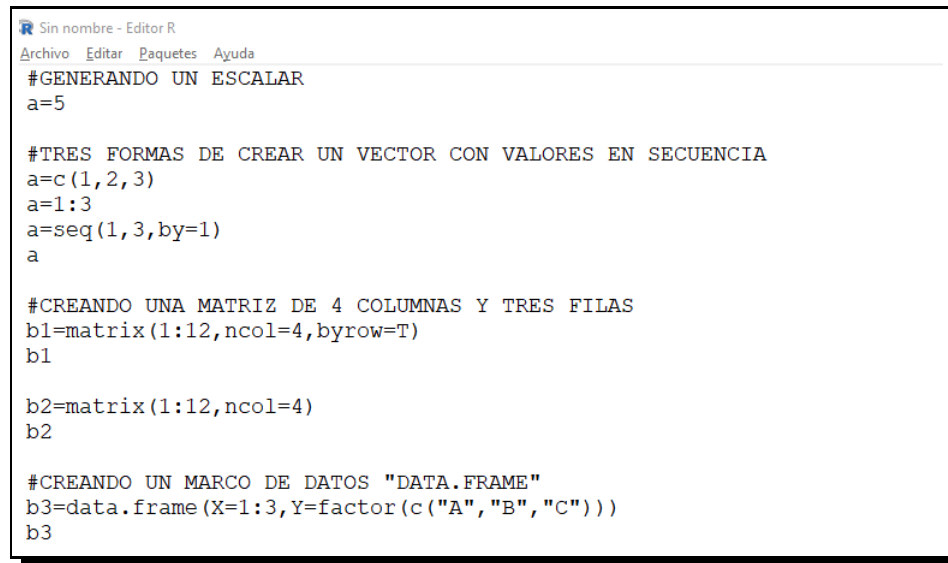
```
A=5
```

```
A<-5
```


En los dos casos se crea un objeto de nombre A de valor numérico o escalar igual a 5. Como R es sensible a mayúsculas, minúsculas y acentos, el objeto llamado A es diferente al objeto llamado a. Para los nombres de objetos se usa una única cadena de caracteres, es decir, no pueden aparecer palabras separadas para denominar un objeto.

1.6.2. Algunos tipos de objetos que se pueden crear en R

Las Figuras 1.14 y 1.15 ilustran la creación de cuatro tipo de objetos R: un escalar, una matriz y un marco de datos o tabla (este objeto admite columnas alfanuméricas).



```
Sin nombre - Editor R
Archivo Editar Paquetes Ayuda

#GENERANDO UN ESCALAR
a=5

#TRES FORMAS DE CREAR UN VECTOR CON VALORES EN SECUENCIA
a=c(1,2,3)
a=1:3
a=seq(1,3,by=1)
a

#CREANDO UNA MATRIZ DE 4 COLUMNAS Y TRES FILAS
b1=matrix(1:12,ncol=4,byrow=T)
b1

b2=matrix(1:12,ncol=4)
b2

#CREANDO UN MARCO DE DATOS "DATA.FRAME"
b3=data.frame(X=1:3,Y=factor(c("A","B","C")) )
b3
```

Figura 1.14: Vista de un Script R donde se editó líneas de programa creando un objeto escalar, un vector numérico, una matriz numérica y un marco de datos

1.6.3. Otros objetos R

Toda función R produce algún resultado, numérico o gráfico, que en sí constituye un objeto R de una clase específica; por ejemplo, la función `ts()` produce un objeto de la clase serie de tiempo, `lm()` produce un objeto de la clase `lm`, etc. La función `class()` sirve para identificar la clase específica de un objeto R, por ejemplo, como se ilustra en Figura 1.16.

1.7. Explicación de algunas funciones de interés

1.7.1. Para lectura de conjuntos de datos

Para la lectura de bases de datos externas usaremos las funciones R `scan()` y `read.table()`. La primera permite leer datos no estructurados desde un archivo o ingresados por teclado; la segunda permite leer datos con estructura tabular bajo el formato de un `data.frame`, guardados en archivos de texto o incluso en excel con formato `.csv`. Existen otras funciones para lectura de datos pero en el curso sólo usaremos estas dos. Veamos cada una.

```

R Console (64-bit)
Archivo  Editar  Misc  Paquetes  Ventanas  Ayuda

> #TRES FORMAS DE CREAR UN VECTOR CON VALORES EN SECUENCIA
> a=c(1,2,3)
> a=1:3
> a=seq(1,3,by=1)
> a
[1] 1 2 3
>
> #CREANDO UNA MATRIZ DE 4 COLUMNAS Y TRES FILAS
> b1=matrix(1:12,ncol=4,byrow=T)
> b1
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
>
> b2=matrix(1:12,ncol=4)
> b2
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
>
> #CREANDO UN MARCO DE DATOS "DATA.FRAME"
> b3=data.frame(X=1:3,Y=factor(c("A","B","C")))
> b3
  X Y
1 1 A
2 2 B
3 3 C
>

```

Figura 1.15: Vista de resultados en consola R de la ejecución de líneas de programa del Script R exhibido en Figura 1.14

```

R Console (64-bit)
Archivo  Editar  Misc  Paquetes  Ventanas  Ayuda

> b=matrix(1:12,ncol=4,byrow=T)
> b
      [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> class(b)
[1] "matrix" "array"
>
> a=1:3
> a
[1] 1 2 3
> class(a)
[1] "integer"
>
> d=3
> d
[1] 3
> class(d)
[1] "numeric"
>

```

```

Sin nombre - Editor R
Archivo  Editar  Paquetes  Ayuda

b=matrix(1:12,ncol=4,byrow=T)
b
class(b)

a=1:3
a
class(a)

d=3
d
class(d)

```

Figura 1.16: Vista ventanas Script R y Consola mostrando líneas de programa creando tres objetos R y el uso de la función class()

read.table()

A continuación se describe su sintaxis invocando los argumentos básicos que se usan en los ejemplos del curso.

```
read.table(file.choose(),header=FALSE,sep=" ",dec=".",skip=0,colClasses=NA)
```

Argumentos:

- `file.choose()`: Habilita la exploración en el sistema de archivos del computador para seleccionar el archivo de datos a ser leído.

- **header:** Un valor lógico (TRUE ó FALSE) para indicar si el archivo contiene los nombres de las variables en la primera línea a ser leída.
- **sep:** Para especificar el carácter que separa los datos en cada línea del archivo. Si se usa `sep=" "`, el separador es un espacio en blanco, uno o más espacios o tabulaciones son casos de este tipo de separador.
- **dec:** El carácter usado en el archivo para los puntos decimales, así `dec="."`, indica que la posición decimal en el archivo leído es indicada por punto y `dec=","` indica que la posición decimal es indicada por coma.
- **skip:** Un valor entero para indicar el número de líneas del archivo de datos que deben saltarse u omitirse desde la primera, antes de comenzar a leer los datos.
- **colClasses:** Este argumento permite determinar mediante un vector de longitud igual al número de columnas en el archivo de datos, la clase de variable a ser asumida en cada caso. Entre los valores posibles para los elementos de este vector, se tienen: "numeric", "factor", "Date", "POSIXct", "NULL". Un valor "NULL" en cualquier posición de tal vector indica que la respectiva columna es omitida en la lectura de los datos, por ejemplo, para un archivo con tres columnas donde la segunda es de valor numérico y la tercera es categórica sería de la siguiente manera, `colClasses=c("NULL","numeric","factor")` hace que se lea sólo la columna 2 como una variable numérica y la columna 3 como un factor, y así el `data.frame` resultante sólo contiene a estas dos variables.

Por ejemplo, para leer los datos en la columna 2 del archivo tipo `.csv` ilustrado en la Figura 1.17 usamos:

```
#Para leer segunda columna en average-weekly-male-earnings-in-editado.csv
datos1=read.table(file.choose(),header=T,skip=8,sep=";",dec=",",colClasses=c("NULL","numeric"))
```

	A	B	C	D	E	F	G	H	I
1	Average weekly male earnings in Australia								
2	Exported from datamarket.com								
3	Date exported	14/07/2012 21:00							
4	On DataMarket	http://datamarket.com/data/set/22xv/#!display=line&ds=22xv12el6=8							
5	License	You are allowed to copy and redistribute the data as long as you clearly indicate the data provider (Time Series Data							
6	Provider	Time Series Data Library (citing: Australian Bureau of Statistics)							
7	Source URL	http://datamarket.com/data/list/?q=provider:tsdl							
8	Units	AUD							
9	Quarter	Averageweeklymaleearnings							
10	1956 q	135,102							
11	1956 q	139,9209							
12	1956 q	137,0656							
13	1956 q	145,1737							
14	1957 q	134,1085							
15	1957 q	140,613							
16	1957 q	140,613							
17	1957 q	148,659							
18	1958 q	134,3511							
19	1958 q	142,9658							
20	1958 q	143,9394							
21	1958 q	151,1278							

Figura 1.17: Vista del contenido del archivo "average-weekly-male-earnings-in-editado.csv"

Nota 1.6. Para determinar cuál es el separador de las columnas de un archivo `.csv` basta abrirlo como texto. Ver en la Figura 1.18 el archivo abierto como texto y note que en cada registro los elementos están separados por punto y coma (;), de ahí que en el ejemplo anterior se usa el argumento `sep=";"` en la función `read.table`; además, la posición decimal corresponde a coma y es necesario saltarse las 8 primeras línea o renglones pues no son parte de los valores a leer.

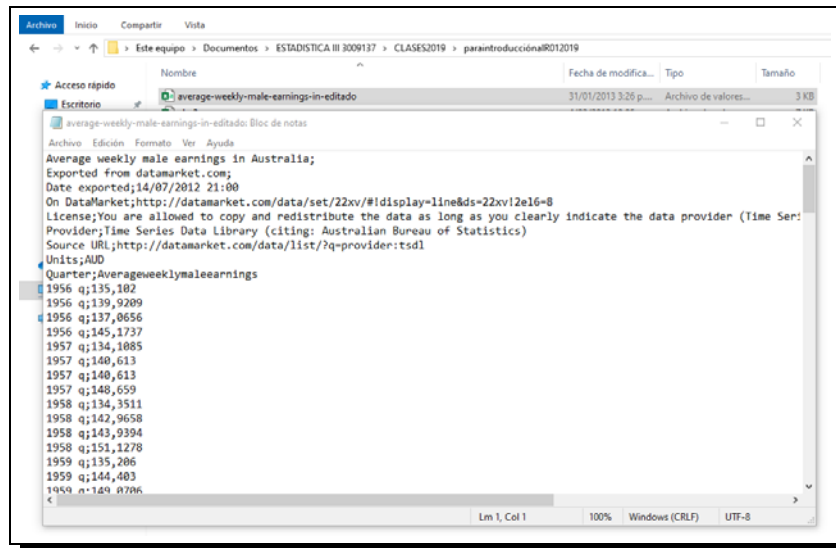


Figura 1.18: Vista del contenido del archivo “average-weekly-male-earnings-in-editado.csv” pero abierto con bloc de notas

scan()

Como previamente se indicó, permite leer datos no estructurados tanto desde la consola como también de archivos externos, aunque también puede adaptarse a la lectura de datos estructurados. La sintaxis básica es la siguiente.

```
scan(file.choose(),sep=" ",dec=".",skip=0,what=double())
```

Argumentos:

- Los argumentos `sep`, `dec` y `skip` se ajustan según sea necesario como se indicó para `read.table()`.
- `what`: Permite especificar el tipo de variables que van a ser leídas o ingresadas en la consola o leyendo archivo externo. Su configuración es mediante una lista en la cual se pueden dar nombre a las variables e indicar su tipo. Por ejemplo, si los datos corresponden a una tabla con dos columnas, que queremos nombrar como `X1` y `X2`, respectivamente, siendo la primera numérica y la segunda alfanumérica, entonces se indicaría esto así: `what=list(X1=0,X2="")`.

Por ejemplo, si se desea ingresar en la consola R un vector con los valores de 200, 150, 500, 234,250, 199, 367, 510, 452, 480, 317, 220, puede hacerse como se ilustra a seguir,

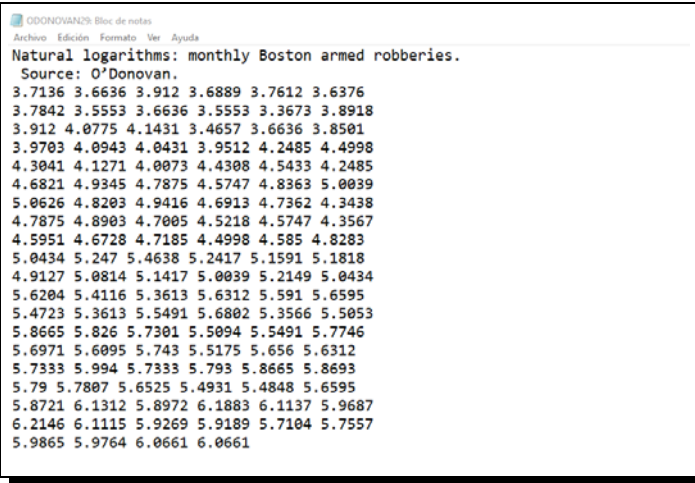
```

datos2=scan()
200 150 500 234
250 199 367 510
452 480 317 220
  
```

En el ejemplo anterior, la función asume sólo un vector de valores y no una tabla de cuatro columnas con tres filas y lee por renglón de izquierda a derecha. Considere a continuación el conjunto de datos en el archivo de texto `ODONOVAN29.txt`, ilustrado en la Figura 1.19 y que corresponden a una serie de tiempo univariada. Para leerlo se procede de la siguiente manera:

```

#Para leer ODOOVAN29.TXT
datos3=scan(file.choose(),dec=".",skip=2)
datos3
  
```



```

ODONOVAN29: Bloc de notes
Archivo  Edición  Formato  Ver  Ayuda
Natural logarithms: monthly Boston armed robberies.
Source: O'Donovan.
3.7136 3.6636 3.912 3.6889 3.7612 3.6376
3.7842 3.5553 3.6636 3.5553 3.3673 3.8918
3.912 4.0775 4.1431 3.4657 3.6636 3.8501
3.9703 4.0943 4.0431 3.9512 4.2485 4.4998
4.3041 4.1271 4.0073 4.4308 4.5433 4.2485
4.6821 4.9345 4.7875 4.5747 4.8363 5.0039
5.0626 4.8203 4.9416 4.6913 4.7362 4.3438
4.7875 4.8903 4.7005 4.5218 4.5747 4.3567
4.5951 4.6728 4.7185 4.4998 4.585 4.8283
5.0434 5.247 5.4638 5.2417 5.1591 5.1818
4.9127 5.0814 5.1417 5.0039 5.2149 5.0434
5.6204 5.4116 5.3613 5.6312 5.591 5.6595
5.4723 5.3613 5.5491 5.6802 5.3566 5.5053
5.8665 5.826 5.7301 5.5094 5.5491 5.7746
5.6971 5.6095 5.743 5.5175 5.656 5.6312
5.7333 5.994 5.7333 5.793 5.8665 5.8693
5.79 5.7807 5.6525 5.4931 5.4848 5.6595
5.8721 6.1312 5.8972 6.1883 6.1137 5.9687
6.2146 6.1115 5.9269 5.9189 5.7104 5.7557
5.9865 5.9764 6.0661 6.0661

```

Figura 1.19: Vista del contenido del archivo “ODONOVAN29.txt”

Nota 1.7. Para leer o ingresar con `scan()` un conjunto de datos estructurados y mantener un formato tabular, es necesario combinar esta función con la función `data.frame`, por ejemplo para leer el archivo “average-weekly-male-earnings-in-editado.csv”, previamente ilustrado en la Figura 1.17, el código es como sigue:

```
#Para leer average-weekly-male-earnings-in-editado.csv
datos2=data.frame(scan(file.choose(),skip=9,sep=";",dec=".",what=list(fecha="",earnings=0)))
```

Note que en el último ejemplo, se ha indicado saltarse los primeros 9 registros, lo cual indica que incluso no se leerá el renglón de los encabezados, y que las dos variables en ese archivo serán leídas y denominadas como `fecha` que es de tipo alfanumérica y `earnings` la cual es numérica.

ts()

Función para crear en R objetos tipo series de tiempo. La sintaxis básica es la siguiente.

```
ts(data,frequency=1,start=c(año,período))
```

Argumentos:

- **data:** Un vector numérico o una matriz que contiene los valores de series de tiempo observadas. Si el objeto es un `data.frame` entonces será transformado a una matriz numérica.
- **start:** El tiempo de inicio de la serie se puede especificar como un vector `c(año,período)` donde `año` es reemplazado por el año y `período` por el período en ese año que corresponde al primer dato de la serie, o simplemente el año para datos anuales.
- **frequency:** El número de observaciones por unidad de tiempo. Por ejemplo, `frequency=12` para datos mensuales, `frequency=4` para datos trimestrales, `frequency=1` para datos anuales.

Por ejemplo, los datos ilustrados en la Figura 1.17 corresponden a una serie de tiempo observada trimestralmente, con fecha inicial trimestre 1 (Q1) de 1956, luego para leer estos datos crear el objeto serie de tiempo, se procede como se muestra a continuación:

```
#Leer segunda columna en average-weekly-male-earnings-in-editado.csv
datos1=read.table(file.choose(),header=T,skip=8,sep=";",dec=".",colClasses=c("NULL","numeric"))
```

```
#Convertir en una serie de tiempo de frecuencia trimestral con fecha de inicio 1956-Q1
datos1=ts(datos1,freq=4,start=c(1956,1))
```

Los datos en el archivo ilustrado en la Figura 1.19, también corresponden a una serie de tiempo, la cual es de frecuencia mensual y tiene fecha de inicio enero de 1980, por tanto la lectura y conversión a objeto serie de tiempo se realiza así:

```
#Para leer ODOONVAN29.TXT
datos3=scan(file.choose(),dec=".",skip=2)
```

```
#Convertir en una serie de tiempo de frecuencia mensual con fecha de inicio 1980-1
datos3=ts(datos3,frequency=12,start=c(1980,1))
```

Un ejemplo más: Lectura de los datos en archivo cbe2.csv que se ilustra en la Figura 1.20 y conversión en serie de tiempo con frecuencia mensual y fecha de inicio enero de 1958:

	A	B	C	D	E	F	G	H	I	J
1	Esto datos the monthly supply of electricity in millions of kwh, beer and chocolate-based productivity in tonnes in Australia,									
2	january 1958 to december 1990									
3	choc	beer	elec							
4	1451	96,3	1497							
5	2037	84,4	1463							
6	2477	91,2	1648							
7	2785	81,9	1595							
8	2994	80,5	1777							
9	2681	70,4	1824							
10	3098	74,8	1994							
11	2708	75,9	1835							
12	2517	86,3	1787							
13	2445	98,7	1699							
14	2087	100,9	1633							
15	1801	113,8	1645							
16	1216	89,8	1597							

Figura 1.20: Vista del contenido del archivo “cbe2.csv”

```
#Para leer cbe2.csv
cbe2.csv=read.table(file.choose(),header=T,dec=',',sep=';',skip=2)
```

```
#Convertir en una serie de tiempo de frecuencia mensual con fecha de inicio 1958-1
cbe2.csv=ts(cbe.csv,freq=12,start=c(1958,1))
```

1.7.2. Creando y extrayendo elementos de un vector en R

Considere el siguiente vector,

```
a=c(1,2,3,4,5,20,21,22,23,24,25)
```

Entonces,

- `a[1]` extrae el primer valor que es 1

- `a[1:5]` extrae los primeros cinco valores que corresponden a 1, 2, 3, 4, 5
- `a[c(1,6,9)]` extrae los valores primero, sexto y noveno, es decir a 1, 20 y 23.

1.7.3. Creando y extrayendo elementos de una matriz o de un `data.frame`

Considere la matriz A,

```
A=matrix(c(1,2,3,4,5,20,21,22,23,24,25,30),ncol=3,byrow=T)
```

Al invocar el anterior objeto R se obtiene como resultado lo siguiente:

```
> A
      [,1] [,2] [,3]
[1,]     1     2     3
[2,]     4     5    20
[3,]    21    22    23
[4,]    24    25    30
```

También podemos crear la anterior matriz con la función `cbind()` que pega vectores o matrices por columnas, así

```
A=cbind(c(1,4,21,24),c(2,5,22,25),c(3,20,23,30))
```

O bien, usando la función `rbind()` la cual pega vectores o matrices por fila,

```
A=rbind(c(1,2,3),c(4,5,20),c(21,22,23),c(24,25,30))
```

Podemos extraer elementos de A, así:

- `A[2,]` extrae la fila 2 de A, y produce por tanto un vector de longitud 3.
- `A[,3]` extrae la columna 3 de A y produce por tanto un vector de longitud 4.
- `A[2,3]` extrae el valor en fila 2 y columna 3, es decir a 20.
- `A[c(1,4),]` extrae las filas 1 y 4 y produce por tanto una matriz de dimensión 2x3.
- `A[,c(1,3)]` extrae las columnas 1 y 3 y produce por tanto una matriz de dimensión 4x2.

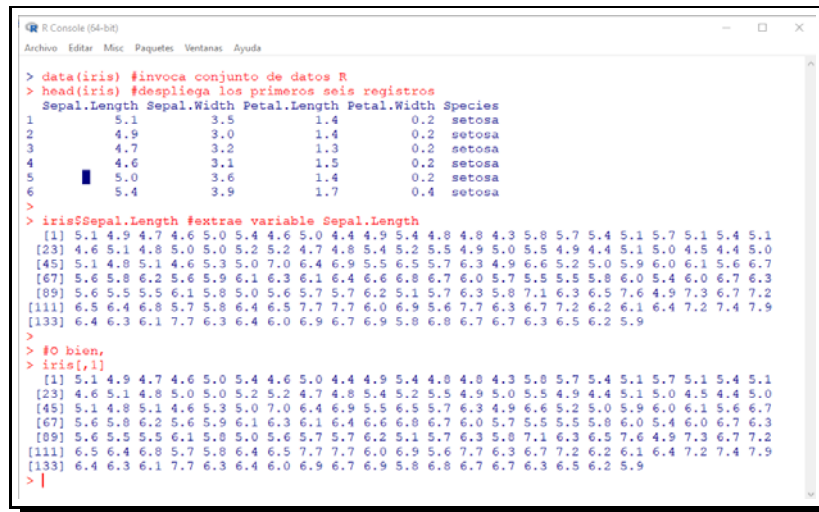
Nota 1.8. Para un objeto tipo `data.frame` también se aplica la extracción de elementos como se hace para el caso de objetos matriciales. Otra manera de extraer variables guardadas como parte de un `data.frame` es como sigue: `nombre-data.frame$nombre-variables`. Por ejemplo,

```
data(iris) #invoca conjunto de datos R
head(iris) #despliega los primeros seis registros

iris$Sepal.Length #extrae variable Sepal.Length

#0 bien,
iris[,1]
```

En la Figura 1.21 se puede ver el resultado de la ejecución del anterior bloque de programación



```

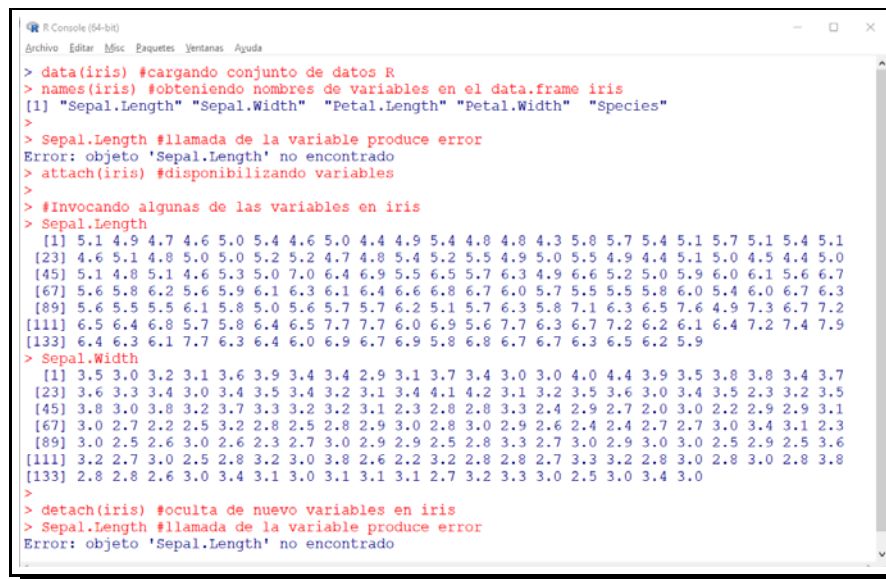
R Console (64-bit)
Archivo  Editar  Mac  Paquetes  Ventanas  Ayuda

> data(iris) #invoca conjunto de datos R
> head(iris) #despliega los primeros seis registros
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1         5.1         3.5         1.4         0.2  setosa
2         4.9         3.0         1.4         0.2  setosa
3         4.7         3.2         1.3         0.2  setosa
4         4.6         3.1         1.5         0.2  setosa
5         5.0         3.6         1.4         0.2  setosa
6         5.4         3.9         1.7         0.4  setosa
>
> iris$Sepal.Length #extrae variable Sepal.Length
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1 5.7 5.1 5.4 5.1
[23] 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0
[45] 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7
[67] 5.6 5.8 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3
[89] 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2
[111] 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9
[133] 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
>
> #O bien,
> iris[,1]
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1 5.7 5.1 5.4 5.1
[23] 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0
[45] 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7
[67] 5.6 5.8 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3
[89] 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2
[111] 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9
[133] 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
> |

```

Figura 1.21: Vista con resultados sobre comandos ejecutados sobre datos R iris

Nota 1.9. Tratándose de un `data.frame`, las variables dentro de ese tipo de objeto pueden ser accedidas mediante `attach(nombre-data.frame)`, este comando disponibiliza las variables que internamente se encuentran dentro del `data.frame` y así estas puedan ser invocadas directamente por su nombre. El proceso anterior puede revertirse usando `detach(nombre-data.frame)`. Vea el ejemplo ilustrado en la Figura 1.22.



```

R Console (64-bit)
Archivo  Editar  Mac  Paquetes  Ventanas  Ayuda

> data(iris) #cargando conjunto de datos R
> names(iris) #obteniendo nombres de variables en el data.frame iris
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
>
> Sepal.Length #llamada de la variable produce error
Error: objeto 'Sepal.Length' no encontrado
> attach(iris) #disponibilizando variables
>
> #Invocando algunas de las variables en iris
> Sepal.Length
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1 5.7 5.1 5.4 5.1
[23] 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0
[45] 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7
[67] 5.6 5.8 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3
[89] 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2
[111] 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9
[133] 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9
> Sepal.Width
[1] 3.5 3.0 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 3.7 3.4 3.0 3.0 4.0 4.4 3.9 3.5 3.8 3.8 3.4 3.7
[23] 3.6 3.3 3.4 3.0 3.4 3.5 3.4 3.2 3.1 3.4 4.1 4.2 3.1 3.2 3.5 3.6 3.0 3.4 3.5 2.3 3.2 3.5
[45] 3.8 3.0 3.8 3.2 3.7 3.3 3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7 2.0 3.0 2.2 2.9 2.9 3.1
[67] 3.0 2.7 2.2 2.5 3.2 2.8 2.5 2.8 2.9 3.0 2.8 3.0 2.9 2.6 2.4 2.4 2.7 2.7 3.0 3.4 3.1 2.3
[89] 3.0 2.5 2.6 3.0 2.6 2.3 2.7 3.0 2.9 2.9 2.5 2.8 3.3 2.7 3.0 2.9 3.0 3.0 2.5 2.9 2.5 3.6
[111] 3.2 2.7 3.0 2.5 2.8 3.2 3.0 3.8 2.6 2.2 3.2 2.8 2.8 2.7 3.3 3.2 2.8 3.0 2.8 3.0 2.8 3.8
[133] 2.8 2.8 2.6 3.0 3.4 3.1 3.0 3.1 3.1 3.1 2.7 3.2 3.3 3.0 2.5 3.0 3.4 3.0
>
> detach(iris) #oculta de nuevo variables en iris
> Sepal.Length #llamada de la variable produce error
Error: objeto 'Sepal.Length' no encontrado

```

Figura 1.22: Ilustración de la funciones `attach()` y `detach()` sobre datos R iris

1.7.4. Gráficas

Los gráficos básicos más usados en el curso son: gráficos de dispersión, histogramas, box plots, gráficos de probabilidad normal. Veamos las funciones R para éstas.

plot()

Función para realizar gráficos de dispersión, de líneas y líneas-punto. Su sintaxis básica es como sigue

```
plot(x, y, ...)
```

Argumentos:

- **x**: El vector con las coordenadas X del par (x,y) de los datos en el gráfico.
- **y**: El vector con las coordenadas Y del par (x,y) de los datos en el gráfico.
- **...**: Demás parámetros gráficos que pueden especificarse, por ejemplo, **main**="título", **xlab**="etiqueta eje X", **ylab**="etiqueta eje Y", **type**="l" para realizar un gráfico donde se une los puntos del gráfico con líneas, **type**="p" para realizar un gráfico de dispersión con puntos sin unir por líneas, **type**="b" para realizar un gráfico donde los datos son representados por puntos unidos por líneas, **xlim**=c(x1,x2) para especificar valor máximo (x2) y mínimo (x1) a presentar en eje X, **ylim**=c(y1,y2) para especificar valor máximo (y2) y mínimo (y1) a presentar en eje Y. Para más argumentos de la función y posibilidades del gráfico consultar **?par**. Todos los argumentos se especifican separados por comas.

Nota 1.10. Aplicar la función **plot(x)** siendo **x** un factor o variable cualitativa, produce un gráfico de barras. **plot(y~x)**, con **x** un factor y **y** un vector numérico produce gráficos de cajas de la distribución de valores de **y** en cada nivel de **x**. Ver el siguiente ejemplo con el conjunto de datos R **iris**.

```
data(iris) #cargando conjunto de datos R
attach(iris) #disponibilizando variables en iris
layout(rbind(c(1,1,2,2),c(3,3,4,4))) #División ventana gráfica y ubicación de
                                     #gráficas en esta división

plot(Sepal.Length,Sepal.Width)
plot(Species,main="plot(Species)")
plot(Sepal.Length,Species,main="plot(Sepal.Length,Species)")
plot(Sepal.Length~Species,main="plot(Sepal.Length~Species)")
detach(iris) #ocultando variables en iris
```

El anterior bloque de programación produce la Figura 1.23.

Nota 1.11. **plot(nombre-data.frame)** produce una matriz de dispersión, por ejemplo aplicado sobre el conjunto de datos R **iris**, produce una matriz de dispersión.

```
data(iris) #cargando conjunto de datos R
plot(iris)
```

En la Figura 1.24 se muestra la gráfica resultante con el bloque de programación.

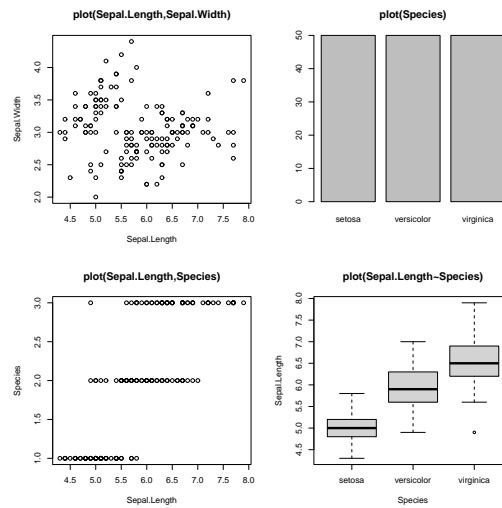


Figura 1.23: Ilustración de la función `plot` sobre variable cuantitativa vs. cuantitativa, sobre variable cualitativa y cuantitativa vs. cualitativa

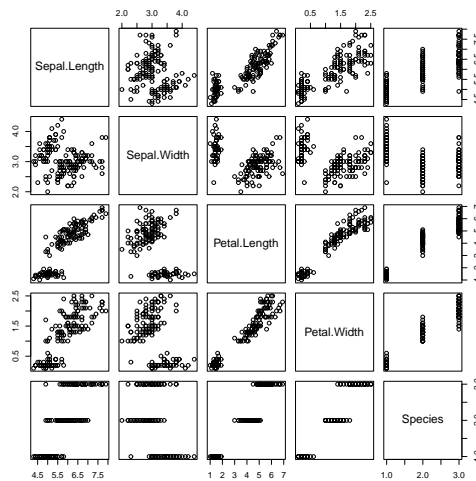


Figura 1.24: Ilustración de la función `plot` sobre el `data.frame iris`

`boxplot()`

Función para realizar gráfico de cajas o box plot. Si se aplica sobre un vector `x` numérico grafica el box plot para el conjunto de datos; si se aplica sobre la fórmula (`y~x`), con `y` un vector numérico y `x` un factor, grafica las box plots (uno al lado del otro) de los valores de `y` en cada nivel de `x`. Esta función admite argumentos gráficos adicionales.

```
boxplot(x, ...)
boxplot(fórmula, ...)
```

Argumentos:

- `x`: Vector numérico.
- `fórmula`: Una fórmula tal como `y~grp`, donde `y` es un vector numérico a ser particionado en grupos de acuerdo a la variable de agrupación `grp` (usualmente un factor).
- `...`: Parámetros gráficos, como por ejemplo, `boxwex` que corresponde al factor de escala a aplicar a todas las cajas permitiendo hacer las cajas más estrechas. `col` para especificar en un vector los colores a aplicar a las cajas.

Los colores en R se definen a través de una cadena de caracteres ("red", "gray", etc.), o con números (1 para color negro, 2 para color rojo, 3 para color verde, etc.). `horizontal=FALSE` las cajas se dibujan verticalmente y `horizontal=TRUE` hace que las cajas se dibujen horizontalmente; `names` para especificar en un vector con valores alfanuméricos los nombres a imprimir debajo de las cajas; `whisklty` para especificar el tipo de línea para los bigotes (ver `lty` en `?par`).

Por ejemplo,

```
data(iris) #cargando conjunto de datos R
attach(iris)
layout(rbind(c(0,1,1,0),c(2,2,3,3)))
boxplot(Sepal.Length,whisklty=1,xlab="Sepal.Length")
boxplot(Sepal.Length~Species,boxwex=0.4,col="orange",horizontal=TRUE,whisklty=1)
boxplot(Sepal.Length~Species,boxwex=0.4,col=c("violet","blue","gray"),horizontal=FALSE,whisklty=1)
```

Del anterior bloque de programa resulta la Figura 1.25.

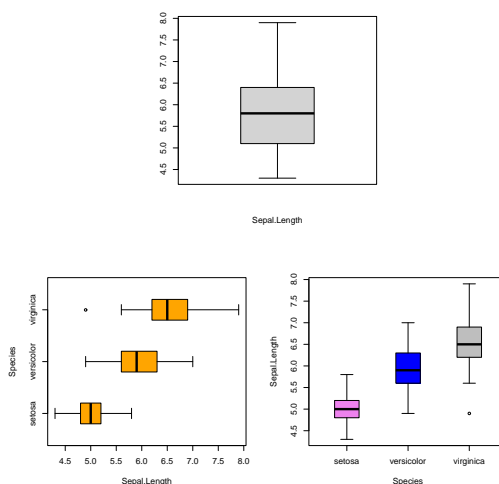


Figura 1.25: Ilustración de la función `boxplot()`

hist()

Función para graficar histogramas de frecuencias (de conteos o de frecuencias relativas)

```
hist(x, ...)
```

Argumentos:

- **x**: Vector numérico con los valores de la variable para la cual se quiere el histograma.
- **...**: Otros parámetros tales como: **breaks** un vector para especificar los puntos de corte entre los intervalos de clase del histograma, o bien, el nombre de una función para calcularlos (por defecto es `breaks="Sturges"`); **freq** argumento lógico (TRUE o FALSE) para indicar si se representarán los conteos de frecuencias absolutas o las frecuencias relativas en cada intervalo de clase; **col** un color a ser usado para rellenar las barras; **density** la densidad de líneas de sombreado en líneas por pulgada.

Ejemplo:

```
data(iris) #cargando conjunto de datos R
attach(iris)
```

```
layout(rbind(c(0,1,1,0),c(2,2,3,3)))
hist(Sepal.Length)
hist(Sepal.Length,col="orange")
hist(Sepal.Length,freq=FALSE,density=10,col="blue")
```

EL anterior bloque de programación produce la Figura 1.26

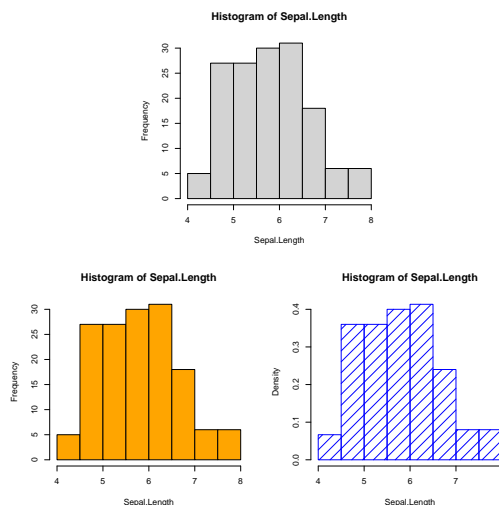


Figura 1.26: Ilustración de la función `hist()`

En el siguiente código observe cómo se agrega a un histograma un boxplot horizontal. Ver la gráfica resultante en la Figura 1.27 (consulte `?par`, los parámetros gráficos `bty`, `xlim`, `ylim`, `xaxt`, `yaxt` usados en este ejemplo en las funciones `hist()`, `boxplot()`).

```
data(iris) #cargando conjunto de datos R
attach(iris)
hist(Sepal.Length,xlim=c(4,8),ylim=c(-8,35),col="white",bty="n")
boxplot(Sepal.Length,bty="n",col="white",xaxt="n",yaxt="n",horizontal=T,boxlwd=2,whisklty=1,whisklwd=3,
        boxwex=10,add=T,at=-5,xlim=c(4,8))
```

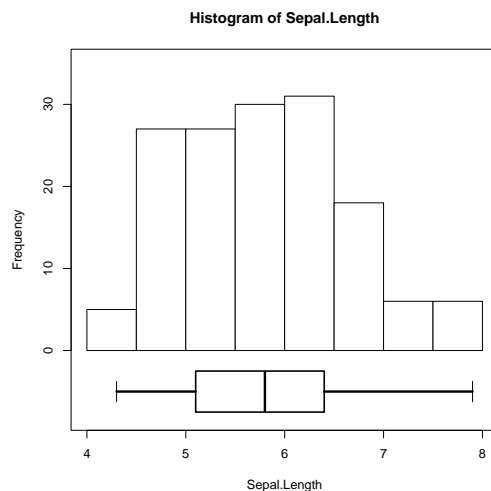


Figura 1.27: Ilustración de un histograma y box plot agregado en parte inferior, usando en la función `boxplot()` los argumentos `horizontal=TRUE` y `add=T`

win.graph()

Función que solo está disponible para R bajo Windows, para abrir una nueva ventana gráfica con el alto y ancho deseado. Se invoca antes de una función graficadora.

```
win.graph(width, height, pointsize)
```

Argumentos:

- **width, height:** Ancho y alto nominal de la ventana gráfica, en pulgadas.
- **pointsize:** Tamaño de los puntos y texto en el gráfico. La unidad aquí representa 1/12 de pulgada.

abline()

Función para agregar líneas rectas (horizontales, verticales o con pendiente) a un gráfico activo.

```
abline(h = NULL, v = NULL,...)
```

Argumentos:

- **h:** Vector de valores para líneas horizontales, es decir paralelas al eje X.
- **v:** Vector de valores para líneas verticales, es decir, paralelas al eje Y.
- **...:** Otros parámetros gráficos tales como **lty=2** (grafica línea punteada), **lwd=2** (grafica una línea de grosor 2 unidades). Por ejemplo, **abline(h=2,col=2)** traza sobre la ventana gráfica activa una línea paralela al eje X, pasando por **y=2**, con color 2 (rojo).

layout()

Función para dividir la ventana gráfica en filas y columnas según la matriz **mat** que se especifique, con el fin de presentar simultáneamente varias figuras en una misma ventana.

```
layout(mat,...)
```

Argumentos:

- **mat:** Una matriz que especifica la ubicación de las siguientes N figuras sobre la ventana gráfica. Cada valor en la matriz puede tomar un valor de 0 o un entero positivo. Los elementos de **mat** corresponden a los números enteros $1, 2, \dots, N$.
- **...** Otros parámetros tales como **widths** que corresponde a un vector para especificar los anchos relativos de las columnas sobre la ventana gráfica y **heights** para especificar con un vector los altos relativos de la filas en la ventana gráfica.

Ejemplo: Colocando cuatro figuras en una ventana gráfica dividida en cuatro celdas, con las dos primeras figuras en la primera fila y las dos últimas en la segunda fila. Los objetos **x1**, **x2**, **x3**, **x4** y **y1**, **y2**, **y3**, **y4** han sido definidos previamente a través de una simulación que no se muestra aquí. Ver Figura 1.28.

```
#Definiendo división en cuatro celdas de igual ancho y alto
matriz=rbind(c(1,1,2,2),c(3,3,4,4))
layout(matriz)
plot(x1,y1)
plot(x2,y2)
plot(x3,y3)
```

```
plot(x4,y4)

layout(matriz,heights=c(1,2)) #partición en cuatro celdas, con fila
                                #inferior 2 veces más alta que la superior

plot(x1,y1)
plot(x2,y2)
plot(x3,y3)
plot(x4,y4)
```

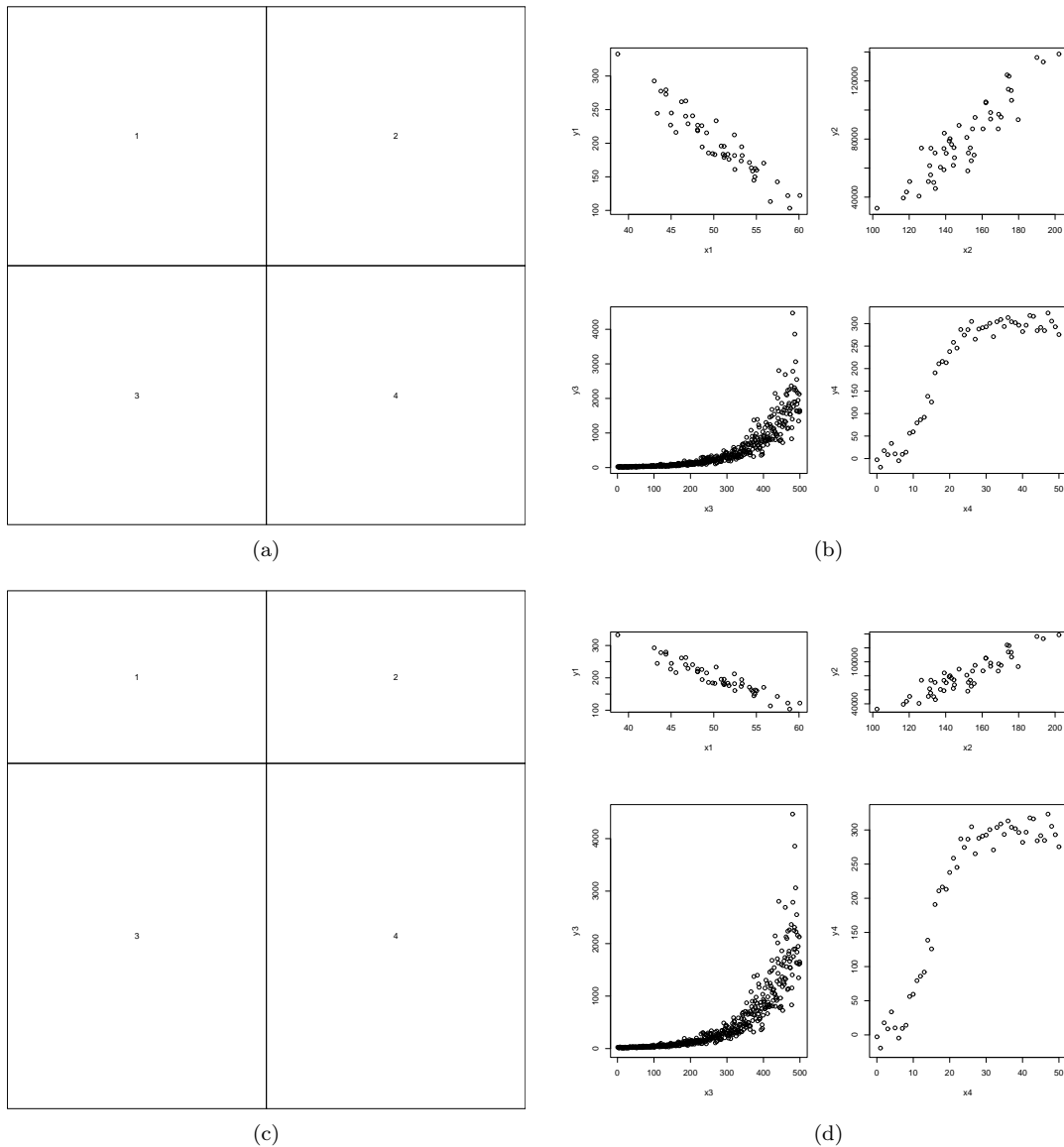


Figura 1.28: Ejemplos de uso de la función `layout()` para particionar una ventana gráfica. Las figuras (a) y (c) muestran cómo queda la partición o diseño de la ventana y las figuras (b) y (d) muestran cómo lucen las gráficas ubicadas en tales particiones. La partición realizada ha usado como argumento `mat` a `matriz=rbind(c(1,1,2,2),c(3,3,4,4))`. En las gráficas inferiores se ha usado además el argumento `heights=c(1,2)`, con el cual el alto de la fila inferior de la partición es 2 veces el alto de la fila superior.

qqnorm(), qqline()

Funciones para gráfico de probabilidad normal. `qqnorm` hace el gráfico de dispersión de los cuantiles muestrales vs. cuantiles normales teóricos, mientras que `qqline` agrega a este gráfico la recta teórica bajo el modelo normal.

```
qqnorm(y,...); qqline(y,...)
```

Argumentos:

- **y**: Vector con los valores muestrales.
- ...: Parámetros gráficos. Para `qqline` sólo parámetros relativos al trazo de líneas, tales como `lwd`, `lty`, `col`. Para `qqnorm` parámetros asociados al tipo de símbolo, color y tamaño para los puntos, así como otros parámetros relativos a ejes, títulos, color de fondo, etc. Ver `?par`.

Por ejemplo

```
y=rnorm(50,10,2) #simulando n=50 obs. de la distrib. N(10,4)
#Gráfico de probabilidad normal
qqnorm(y,cex=1.2,main="Gráfico de probabilidad normal para una muestra de n=50 obs. de N(10,4)")
qqline(y, col=2,lwd=2)
```

La gráfica resultante se muestra en la Figura 1.29.

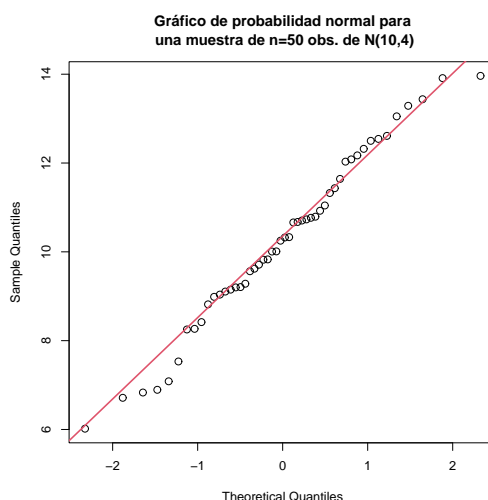


Figura 1.29: Ilustración de un gráfico de probabilidad normal, usando funciones `qqnorm()`, `qqline()`

decompose()

Función que tiene implementado el filtro llamado de descomposición clásica, que permite estimar de manera no paramétrica las componentes de tendencia, estacionalidad y aleatoria, bajo una descomposición aditiva o multiplicativa. Su sintáxis básica es como sigue:

```
decompose(x, type = c("additive", "multiplicative"))
```

Argumentos:

- **x**: Un objeto serie de tiempo, mensual o trimestral.
- **type**: Tipo de descomposición. Por defecto es aditiva.

Para obtener la gráfica de la descomposición o de alguna de las componentes, usamos esta función dentro de `plot()`. Por ejemplo,

```
#Leer segunda columna en average-weekly-male-earnings-in-editado.csv
datos1=read.table(file.choose(),header=T,skip=8,sep=";",dec=".",colClasses=c("NULL","numeric"))
```

```
#Convertir en una serie de tiempo de frecuencia trimestral
#con fecha de inicio 1956-Q1
datos1=ts(datos1,freq=4,start=c(1956,1))

#Gráfica de la descomposición aditiva
plot(decompose(datos1))

#Gráfica de la descomposición multiplicativa
plot(decompose(datos1,type="multiplicative"))

#Gráfica de la estimación de tendencia filtrada por descomposición aditiva
plot(decompose(datos1)$trend,ylim=c(min(datos1),max(datos1)))

#Gráfica de la estimación de la estacionalidad filtrada por descomposición aditiva
plot(decompose(datos1)$seasonal)

#Gráfica del residual causado en la descomposición aditiva
plot(decompose(datos1)$random,ylab="random")
```

Ver en Figura 1.30 resultados gráficos de este ejemplo.

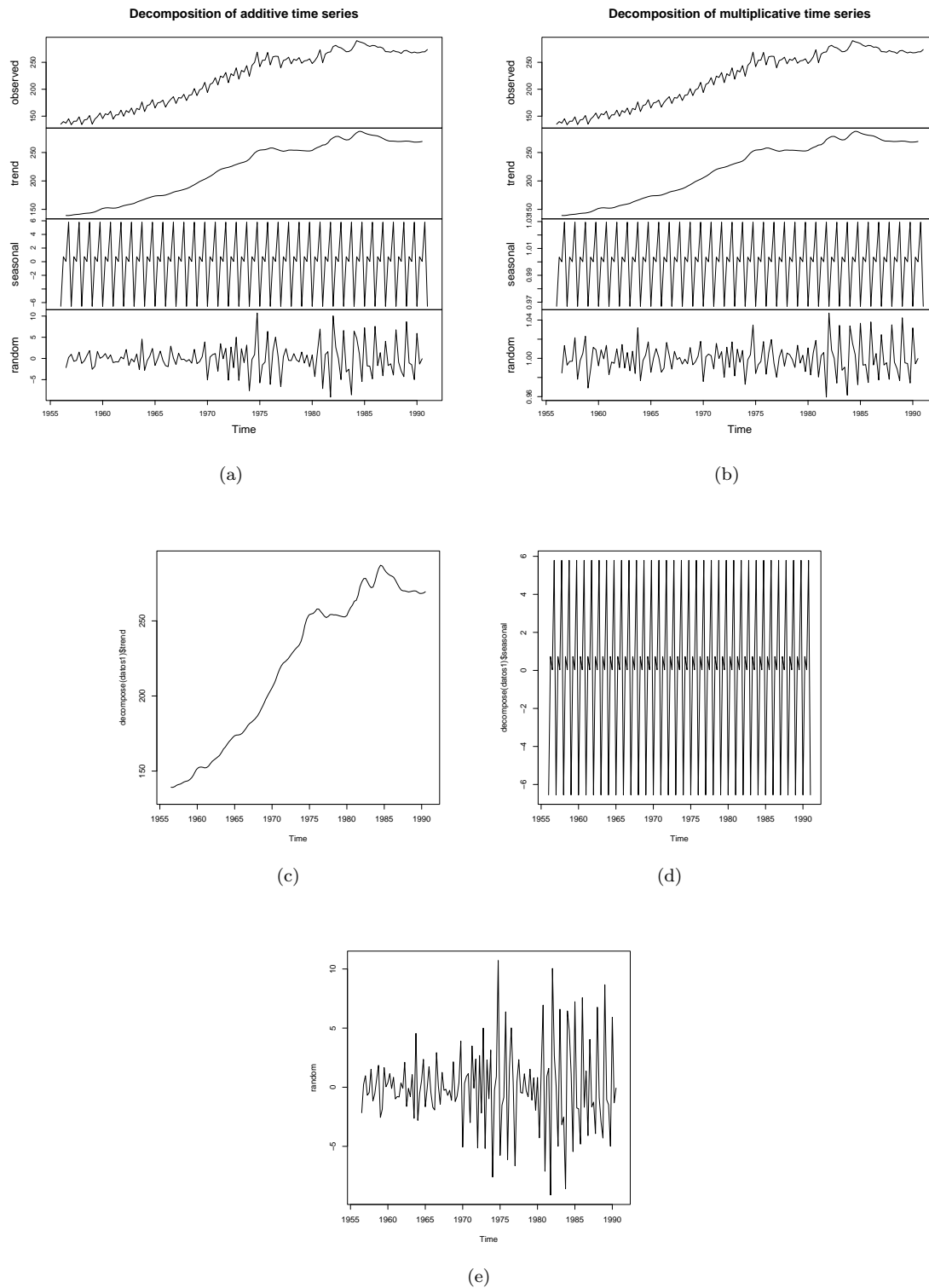


Figura 1.30: Ejemplos de uso de la función `decompose()` para filtrar componentes en una serie de tiempo. Las figuras (a) y (b) muestran las descomposiciones aditiva y multiplicativa, respectivamente; las figuras (c), (d) y (e), muestran por separado las componentes de tendencia, estacionalidad y residual, de la descomposición aditiva.

Bibliografía

- R Core Team (2024), *R: A Language and Environment for Statistical Computing. Reference Index, Version 4.4.2 (2024-10-31)*. R Foundation for Statistical Computing. <https://cran.r-project.org/>
- Venables, W.N., Smith, D.M., and , the R Core Team (2024), *An Introduction to R: Notes on R: A Programming Environment for Data Analysis and Graphics, Version 4.4.2 (2024-10-31)*. <https://cran.r-project.org/>