

# Sesión 2: Introducción a Rstudio y su lenguaje de programación.

Jose Rodney Menezes De la Cruz.



[rodney.menezes@pucp.edu.pe](mailto:rodney.menezes@pucp.edu.pe)

Taller: Rstudio Aplicado a Finanzas  
Colegio de Economistas de Loreto

CELOR, 2019

# Índice

1. Descripción del Programa
2. Interfaz de RStudio
3. Introducción a lo Básico
4. Vectores
5. Matrices
6. Factores
7. Directorio de Trabajo
8. Data Frame
9. Listas

# DESCRIPCIÓN DEL PROGRAMA

# Descripción del programa

- RStudio es una interfaz que permite acceder de manera sencilla a toda la potencia de R.
- Para utilizar RStudio se requiere haber instalado R previamente.
- R es un lenguaje orientado a objetos, es un lenguaje para el cálculo estadístico y la generación de gráficos, que ofrece una gran variedad de técnicas estadísticas, econométricas y gráficas.

# Descripción del programa

- Es un lenguaje de programación completo con el que se añaden nuevas técnicas mediante la definición de funciones.
- Aunque comenzar a trabajar con R es más complejo que hacerlo con programas como Stata, SPSS, etc; sin embargo, tiene muchas ventajas sobre ellos.
- Una de las más importantes es que es un software libre en el que colaboran muchos usuarios para ampliar sus funciones.
- Se encuentra disponible en <https://cran.r-project.org/> y <https://www.rstudio.com/products/rstudio/#Desktop>

# Comparación con otros programas

**Cuadro:** Comparación con otros programas

	Stata	SPSS	SAS	R
Curva de aprendizaje	Empinada	Plana	Poco más de empinada	Muy empinada
Interfaz del usuario	Programación	Interactiva	Programación	Programación
Manipulación de datos	Poderoso	Moderado	Poderoso	Poderoso
Análisis de datos	Poderoso	Poderoso	Versátil	Poderoso
Gráficos	Muy buenos	Muy buenos	Buenos	Muy buenos
Costo	Caro	Muy caro	Muy caro	Gratis

# Funciones de RStudio

- RStudio es una herramienta IDE (Integrated Development Environment) para R, libre y gratuita que facilita:
  - Trabajar con R y gráficos de R de forma interactiva.
  - Organizar el código y mantener múltiples proyectos
  - Mantenimiento de los paquetes de R.
  - Crear y compartir informes.
  - Compartir códigos y colaborar con otros usuarios

# Páginas para aprender y consultar

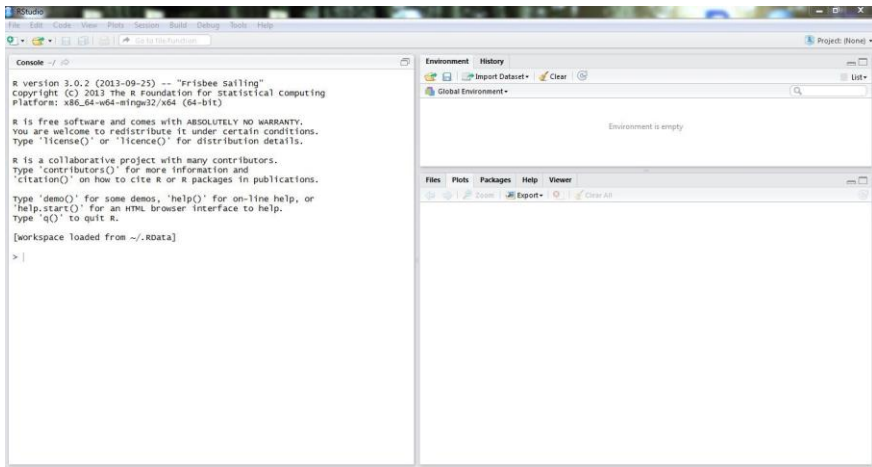
- Totalmente recomendable esta página para aprender R, Python y SQL
  - <https://www.datacamp.com/home>
- Para consultas y resolución de dudas de procedimientos
  - <https://stackoverflow.com/>
- Para consultas avanzadas:
  - <https://github.com/>






# INTERFAZ DE RSTUDIO

# Interfaz de RStudio

- Al ejecutar el programa RStudio se genera la siguiente pantalla:

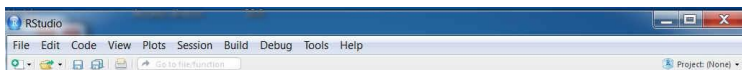


# Interfaz de RStudio

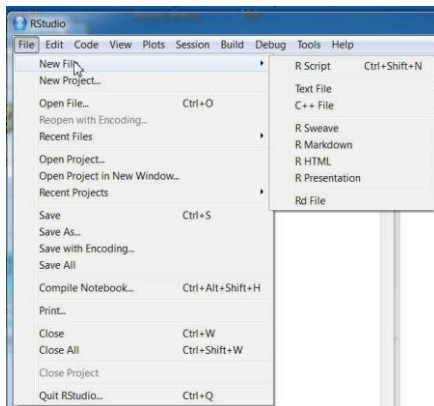
- Esta pantalla está dividida en tres partes:
- La ventana de la izquierda donde está el prompt “>”, llamada Consola, es el espacio de trabajo.
- La ventana de la derecha se divide en dos:
  - En la ventana superior derecha se encuentra el historial de objetos almacenados en memoria. Desde esta ventana también podemos:
    - Limpiar nuestro historial 
    - Importar datos 
    - Muestra los comandos y funciones implementadas de los informes con los que se han trabajado 
  - En la ventana inferior de la derecha RStudio muestra el directorio de trabajo, los gráficos que se van generando, paquetes para cargarlos e instalarlos directamente, ayuda y un visor HTML. Estas pestañas se irán describiendo a lo largo del documento.

## Barra del menú principal: Opciones

- Desde la barra del Menú principal se puede acceder a todos los menús de RStudio. Los primeros menús: Archivo, Edición, Ver y Ayuda son habituales en los programas bajo Windows. El resto de menús son específicos de RStudio estos permiten realizar cambios en los datos, obtener resultados estadísticos, numéricos, gráficos.



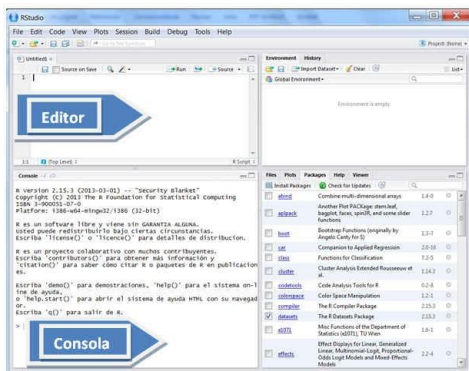
# El Menú File



- Este menú contiene las opciones más generales que suelen tener todos los programas como abrir un archivo, guardar, cerrar, etc.

# El Menú File

- Una de las principales características de RStudio es la flexibilidad para trabajar con diferentes archivos que podemos generar desde New file.
  - New File -> R Script: permite abrir una nueva ventana en la interfaz de RStudio, la ventana de Edición. Aunque se puede trabajar en la consola, no es la forma más eficiente de trabajar en RStudio.



# El Menú File

- En la ventana de edición se escriben las instrucciones y para ejecutar estas instrucciones se pulsa (Ctrl + Enter); (Ctrl + r) o pinchando en el icono Run.
  - New File -> Text File: Permite abrir la venta de texto, en los archivos tipo texto no se pueden ejecutar ninguna función a menos que se copien y se peguen en el espacio de trabajo.
  - New File -> C++ File: Permite compilar funciones de C++ en R.
  - New File -> R Sweave: Crea un archivo que permite trabajar con LaTeX.
  - New File -> R Markdown y New File -> R HTML: Herramienta de RStudio para la creación de informes web. Markdown es un lenguaje simple de marcas diseñado para hacer que el contenido web sea fácil. En lugar de escribir el código HTML y CSS, Markdown permite el uso de una sintaxis mucho más cómoda.

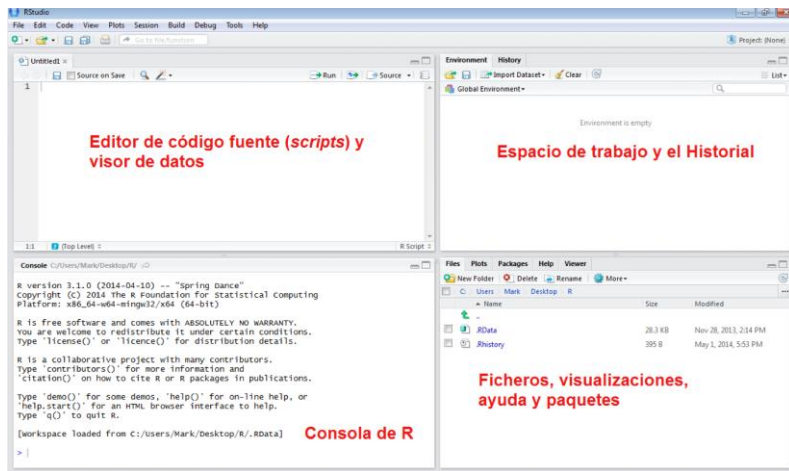
# El Menú File

- Otras opciones:

- New File -> R Presentation: Herramientas de RStudio para hacer presentaciones sencillas. El objetivo de las presentaciones es hacer diapositivas que hacen uso del código R y ecuaciones LaTeX lo más sencillo posible.
- New File -> Rd File: Uno de los requisitos básicos para los paquetes de R es que todas las funciones exportadas, objetos y conjuntos de datos tienen la documentación completa. RStudio también incluye un amplio soporte para la edición de documentación de R, (Los archivos Rd utilizan un formato simple de marcas que se asemeja sintácticamente LaTeX).
- New Project: Permite crear proyectos que hacen que sea más fácil dividir el trabajo en múltiples contextos, cada uno con su propio directorio de trabajo, espacio de trabajo, historial y los documentos de origen. Los proyectos de RStudio están asociados a los directorios de trabajo.



# Entorno de RStudio



# ¿Cómo Funciona RStudio?

- Editor de código fuente (scripts) y visor de datos: permite editar código fuente R y ver los datos del tipo `data.frame`.
- Espacio de trabajo e historial: muestra los objetos (datos/variables) usados en la sesión actual y el historial de comandos.
- Consola R: permite trabajar con R directamente.
- Ficheros, visualizaciones, ayuda y paquetes: permite navegar por los ficheros y carpetas, mostrar los gráficos y visualizaciones, usar la ayuda, e instalar paquetes o cargar paquetes ya instalados. Todos los paneles permiten ser minimizados o maximizados, como ventanas normales de Windows.

# INTRODUCCIÓN A LO BÁSICO

# Instalación de Paquetes y Opciones de Ayuda

- La ayuda se puede utilizar para obtener referencias de cualquier comando que se encuentre disponible en la librería, por ejemplo:

```
help(lm)  
?(lm)
```

- Para instalar paquetes es necesario conexión a internet:

```
install.packages("plm")  
help(plm)
```

# Comentarios en RStudio

- R hace uso del signo # para agregar comentarios, de modo que usted y otros puedan entender de qué se trata el código R.
- Si usted desea colocar todo un párrafo como comentario, deberá presionar las combinaciones CTRL + SHIFT + C en Windows o CMD + SHIFT + C en MAC.

# Aritmética con R

- Adicción: +
- Resta: -
- Multiplicación: \*
- División: /
- Exponencia: ^
- Modulo: %
  - Este último devuelve el resto de la división del número a la izquierda por el número a su derecha, por ejemplo 5 módulo 3 o `5 % 3` es 2.

# Asignación a variable

- Un concepto básico en programación estadística es llamar a una variable.
- Una variable permite guardar un valor (ejemplo 4) o un objeto (ejemplo una función) en R.
- De esta manera, luego puedes usar el nombre de esta variable para acceder a su valor o al objeto que es guardado dentro de esta variable usando el comando `print`

```
nota <- 8  
nota = 8  
print(nota)
```

# Tipo de Datos en R

- R trabaja con numerosos tipos de datos. Algunos de los más básicos tipos son los siguientes:
  - Valores decimales como 4.5 son llamados numerics
  - Numeros naturales como 4 son llamados integers (enteros). Integers son también numéricos.
  - Los valores TRUE o FALSE son llamados logical
  - Valores de texto (o string) son llamados characters

## Nota

Las comillas indican que "algún texto" es un carácter.



# Tipo de Datos en R

Type	Test	Conversion
character	is.character	as.character
complex	is.complex	as.complex
double	is.double	as.double
expression	is.expression	as.expression
integer	is.integer	as.integer
list	is.list	as.list
logical	is.logical	as.logical
numerical	is.numeric	as.numeric
single	is.single	as.single
raw	is.raw	as.raw

# VECTORES

# Vectores

- Los vectores son matrices de una dimensión que pueden contener datos numéricos, datos de caracteres o datos lógicos.
- En otras palabras, un vector es una herramienta simple para almacenar datos. Por ejemplo, puede almacenar sus ganancias y pérdidas diarias durante un mes.
- En R, creas un vector con la función de combinación `c()`. Coloca los elementos del vector separados por una coma entre los paréntesis. Por ejemplo:

```
numeric_vector <- c(1, 2, 3)
character_vector <- c("a", "b", "c")
```

- Una vez creado estos vectores en R, se puede usar para hacer calculos u otros aplicaciones.

# Nombrando un Vector

- Tu puedes nombrar los elementos de un vector con la función `names`

```
some_vector <- c("Ronald", "Economista")  
names(some_vector) <- c("Nombre", "Profesion")
```

- Este código crea `some_vector` y luego da a los dos elementos un nombre.

# Calculando Ingresos Totales

- Puedes realizar cálculos a través de vectores:

```
mes1 <- c(1, 2, 3)
mes2 <- c(4, 5, 6)
total_meses <- a +
b
print(total_meses)
5, 7, 9
```

- Una función que ayuda para calcular la suma total de todos los elementos de un vector es `sum()`.

```
sum(total_meses)
21
```

# Selección de elementos dentro de un Vector

- Para seleccionar elementos de un vector, se hace uso de corchetes [ ]. Entre estos corchetes, tu indicas los elementos a seleccionar.
- Por ejemplo, seleccionamos el segundo elemento de un vector:

```
total_meses[2]
```

- Para seleccionar multiples elementos de un vector, indicas entre los corchetes que elementos debe ser seleccionado:

```
total_meses[c(1, 3)]
```

- En caso tengas 10 elementos dentro de un vector y quieres seleccionar desde el segundo elemento hasta el sexto elemento:

```
total_meses[2:6]
```

# Selección por comparación

- Para comparaciones, usamos los siguientes operadores

< para menor que

> para mayor que

<= para menor o igual que

>= para mayor o igual que

== para igual a otro

!= no igual a otro

- Un ejemplo es:

```
c(4, 5, 6) > 5
```

```
[1] FALSE FALSE TRUE
```

# MATRICES



# ¿Qué es una matriz?

- En R, una matriz es una colección de elementos del mismo tipo de datos (numéricos, de caracteres o lógicos) organizados en un número fijo de filas y columnas. Como solo está trabajando con filas y columnas, una matriz se denomina bidimensional.
- Tu puedes construir una matriz con la función `matrix()`:

```
matrix(1:9, byrow = TRUE, nrow = 3)
```

- En la función `matrix()`:
  - El primer argumento es la colección de elementos que R organizará en las filas y columnas de la matriz. Aquí, usamos los elementos del 1:9.
  - El argumento `byrow` indica que la matriz está llena por las filas. Si queremos que la matriz se llene por las columnas, simplemente colocamos `byrow = FALSE`.
  - El tercer argumento `nrow` señala que la matriz debe tener tres filas.

# Nombrando una matriz

- Similar a los vectores, agregamos nombres para las filas y las columnas de una matriz

```
rownames(my_matrix) <- row_names_vector  
colnames(my_matrix) <- col_names_vector
```

# Calculos con matrices

- La función `rowSums()` convenientemente calcula los totales para cada fila de una matriz.

```
rowSums(my_matrix)
```

- La función `colSums()` convenientemente calcula los totales para cada columna de una matriz.

```
colSums(my_matrix)
```

# Agregando una columna y una fila

- Tu puedes agregar una columna o múltiple columnas a una matriz con la función `cbind()`, el cual une matrices y/o vectores juntos por columna.

```
big_matrix <- cbind(matrix1, matrix2, vector1 ...)
```

- También se puede agregar filas o múltiples filas con la función `rbind()`

# Aritmética con matrices

- Similar a vectores, las operaciones  $+$ ,  $-$ ,  $/$ ,  $*$ , etc. trabajan también con matrices en R.
- Por ejemplo, `2*my_matrix` multiplica cada elemento por dos.

## Nota

En R, para la multiplicación de matrices se debe usar `% * %`

# FACTORES

# ¿Qué es un factor y por qué lo usarías?

- El término factor se refiere a un tipo de datos estadísticos utilizado para almacenar variables categóricas. La diferencia entre una variable categórica y una variable continua es que una variable categórica puede pertenecer a un número limitado de categorías. Una variable continua, por otro lado, puede corresponder a un número infinito de valores.
- Es importante que R sepa si se trata de una variable continua o categórica, ya que los modelos estadísticos que desarrollará en el futuro tratan a ambos tipos de manera diferente.
- Un buen ejemplo de una variable categórica es el sexo. En muchas circunstancias, puede limitar las categorías de sexo a “Masculino” o “Femenino”. (En ocasiones, es posible que necesite diferentes categorías. Por ejemplo, es posible que deba considerar el tipo de combustible, medidas de bienestar subjetivo, diferentes normas culturales, pero siempre tendrá un número finito de categorías).

# ¿Qué es un factor y por qué lo usarías?

- Para crear factores en R, utiliza la función `factor()`. Lo primero que debe hacer es crear un vector que contenga todas las observaciones que pertenecen a un número limitado de categorías.
- Por ejemplo, `sexo_vector` contiene el sexo de 5 individuos diferentes:

```
sexo_vector <- c("Hombre", "Mujer", "Mujer", "Hombre", "Hombre")
```

- Es claro que hay dos categorías, o en términos de R “factor levels”, aquí: “Hombre” y “Mujer”
- La función `factor()` codificará el vector como un factor:

```
factor_sexo_vector <- factor(sexo_vector)
```



# ¿Qué es un factor y por qué lo usarías?

- Hay dos tipos de variables categóricas: una variable categórica nominal y una variable categorica ordenada.
- Una variable nominal es una variable categórica que no implica un orden. Por ejemplo:

```
animales_vector <- c("Elefante", "Jirafa", "Mono", "Caballo")
```

- En contraste, variables ordenadas siguen un orden natural. Por ejemplo:

```
temperatura_vector <- c("Alta", "Baja", "Alta", "Baja", "Media")
```

# Factores Ordenados

- La función `factor()` transforma un vector en un factor no ordenado.
- Para crear un factor ordenado, tu debes agregar dos argumentos adicionales: `ordered` y `levels`.

```
factor(some_vector, ordered = TRUE, levels = c("lev1", "lev2", ...))
```

- Si colocamos `TRUE` a `ordered` en la función `factor()`, tu indicas que el factor está ordenado.
- Con el argumento `levels` tu das los valores del factor en el orden correcto.

# DIRECTORIO DE TRABAJO

# Directorio de Trabajo

- Para conocer el actual directorio de trabajo donde se está trabajando, escribir:

```
getwd()
```

- Para cambiar el directorio de trabajo:

```
setwd("RStudio/Sesión 1/Inputs")
```

- Es importante colocar en la ruta el simbolo de / y no \ sea Windows o Mac.

# Organización del Directorio de Trabajo

- Es importante organizar el Directorio de Trabajo cuando se trabaja con bases de datos.
- Por ello, recomiendo siempre tener las siguientes carpetas para cualquier tipo de trabajo que realicen:
  - Carpeta “Inputs”: aquí se colocan los archivos originales
  - Carpeta “Works”: aquí se colocan todos los procesos y modificaciones que se harán a los archivos originales.
  - Carpeta “Scripts”: aquí se colocan todos los scripts de programación que sea necesarios
  - Carpeta “Outputs”: aquí se colocan todos los resultados o productos finales de sus bases de datos.
  - Carpeta “Graphs”: aquí se colocan todos los gráficos

# Organización del Directorio de Trabajo

- Por ejemplo, tenemos la siguiente carpeta:

```
inputs  <- setwd("/RStudio/Sesión 1/inputs/")
works   <- setwd("/RStudio/Sesión 1/works/")
scripts <- setwd("/RStudio/Sesión 1/scripts/")
outputs <- setwd("/RStudio/Sesión 1/outputs/")
graphs  <- setwd("/RStudio/Sesión 1/graphs/")
```

- Para poder usar estos directorios, usamos lo siguiente:

```
base_excel <- read.xls(paste0(inputs, "base_excel.xls"))
saveRDS(base_excel, paste0(works, "base.rds"))
base_excel <- readRDS(paste0(works, "base.rds"))
```

# DATA FRAME

# ¿Qué es un Data Frame?

- Cuando realiza una encuesta de investigación de mercado, a menudo tiene preguntas como:
- “¿Estás casado?” o preguntas “sí / no” (lógicas) “¿Cuántos años tienes?” (numérico) “¿Cuál es su opinión sobre este producto?” u otras preguntas abiertas (carácter) ... El resultado, es decir, las respuestas de los encuestados a las preguntas formuladas anteriormente, es un conjunto de datos de diferentes tipos de datos. A menudo se encontrará trabajando con conjuntos de datos que contienen diferentes tipos de datos en lugar de uno solo.
- Un data frame tiene las variables de un conjunto de datos como columnas y las observaciones como filas. Este será un concepto familiar para aquellos que vienen de diferentes paquetes de software estadístico, como SAS o SPSS.



# Head and Tail

- Trabajar con grandes conjuntos de datos no es infrecuente en el análisis de datos. Cuando trabaja con conjuntos de datos y data frames (extremadamente) grandes, su primera tarea como analista de datos es desarrollar una comprensión clara de su estructura y elementos principales. Por lo tanto, a menudo es útil mostrar solo una pequeña parte de todo el conjunto de datos.
- Entonces, ¿cómo hacer esto en R? Bueno, la función `head()` le permite mostrar las primeras observaciones de un marco de datos. De manera similar, la función `tail()` muestra las últimas observaciones en su conjunto de datos.
- Tanto `head()` como `tail()` muestran una línea superior llamada “encabezado”, que contiene los nombres de las diferentes variables en su conjunto de datos.

# Selección de Data Frame

- De forma similar a los vectores y matrices, seleccionar elementos de un data frame se hace con la ayuda de corchetes []. Al usar una coma, puede indicar qué seleccionar de las filas y las columnas respectivamente. Por ejemplo:
  - `base2017[1, 2]` selecciona el valor en la primera fila y la segunda columna en `base2017`.
  - `base2017[1: 3, 2: 4]` selecciona las filas 1, 2, 3 y las columnas 2, 3, 4 en `base2017`.
- A veces desea seleccionar todos los elementos de una fila o columna. Por ejemplo, `base2017[1,]` selecciona todos los elementos de la primera fila.

# Selección de Data Frame

- En vez de usar numéricos para seleccionar elementos de un data frame, se puede usar el nombre de las variables para seleccionar columnas de un data frame.
- Suponer que queremos seleccionar los tres primeros elementos de la variable provincia. Uno puede hacer esto

```
base2017[1:3, 1]
```

- Sin embargo, una manera más fácil es:

```
base2017[1:3, "provincia"]
```

- Si queremos seleccionar una columna a través de su nombre, debemos usar el signo \$

```
base2017$provincia
```

# Selección de Data Frame

- Ahora, utilicemos la función `subset()`.

```
subset(base2017, subset = some_condition)
```

- El primer argumento de `subset()` especifica el conjunto de datos para el que desea un subconjunto.
- Al agregar el segundo argumento, le da a R la información y las condiciones necesarias para seleccionar el subconjunto correcto.

# Sorting (Ordenamiento)

- En el análisis de datos, puede ordenar sus datos según una determinada variable en el conjunto de datos.
- En R, esto se hace con la ayuda de la función `order()`.
- `order()` es una función que le da la posición clasificada de cada elemento cuando se aplica en una variable.

# LISTAS

# Listas, ¿por qué las necesitarías?

- Hemos aprendido hasta aquí:
- Vectores (matriz unidimensional): puede contener valores numéricos, de caracteres o lógicos. Todos los elementos de un vector tienen el mismo tipo de datos.
- Matrices (matriz bidimensional): pueden contener valores numéricos, de caracteres o lógicos. Todos los elementos en una matriz tienen el mismo tipo de datos.
- Data Frames (objetos bidimensionales): pueden contener valores numéricos, de caracteres o lógicos. Dentro de una columna, todos los elementos tienen el mismo tipo de datos, pero diferentes columnas pueden ser de diferentes tipos de datos.

# Listas, ¿por qué las necesitarías?

- Una lista en R es similar a su lista de tareas pendientes en el trabajo o la escuela: los diferentes elementos en esa lista probablemente difieran en longitud, características y tipo de actividad que se debe realizar.
- Una lista en R le permite reunir una variedad de objetos bajo un nombre (es decir, el nombre de la lista) de una manera ordenada. Estos objetos pueden ser matrices, vectores, marcos de datos, incluso otras listas, etc. Ni siquiera es necesario que estos objetos estén relacionados entre sí de ninguna manera.
- Podría decir que una lista es un tipo de súper tipo de datos: ¡puede almacenar prácticamente cualquier información en ella!



# Creando una Lista

- ¡Creemos nuestra primera lista! Para construir una lista usas la función `list()`:

```
my_list <- list(comp1, comp2 ...)
```

- Los argumentos de la función `list` son los componentes de la lista.
- Recuerda, estos componentes pueden ser matrices, vectores, otras listas, ...

# Nombrando una Lista

- Al igual que en su lista de tareas pendientes, usted desea evitar no saber ni recordar qué significan los componentes de su lista. Es por eso que debes darles nombres:

```
my_list <- list(name1 = your_comp1,  
                name2 = your_comp2)
```

- Esto crea una lista con componentes que se denominan name1, nombre2, etc.
- Si desea nombrar sus listas después de haberlas creado, puede usar la función names() como lo hizo con los vectores.
- Los siguientes comandos son totalmente equivalentes a la asignación anterior:

```
my_list <- list(your_comp1, your_comp2)  
names(my_list) <- c("name1", "name2")
```

# Selección de una Lista

- Su lista a menudo se construirá a partir de numerosos elementos y componentes. Por lo tanto, obtener un único elemento, varios elementos o un componente de él no siempre es sencillo.
- Una forma de seleccionar un componente es usar la posición numerada de ese componente. Por ejemplo, para "agarrar" el primer componente `demy_list` se tipea

```
my_list[[1]]
```

- Importante recordar: para seleccionar elementos de vectores, utilice corchetes simples: `[ ]`.
- También puede referirse a los nombres de los componentes, con `[ [ ] ]` o con el signo `$`. Ambos seleccionarán el data frame que representa las revisiones:

```
my_list$df
```

```
my_list [["zona"]]
```

# Agregando Información a la Lista

- Para agregar elementos a las listas de manera conveniente, puede usar la función `c()`, que también se usa para construir vectores:

```
ext_list <- c (my_list, my_val)
```

- Esto simplemente extenderá la lista original, `my_list`, con el componente `my_val`. Este componente se añade al final de la lista. Si desea dar un nombre al nuevo elemento de la lista, simplemente agregue el nombre como lo hizo antes:

```
ext_list <- c (my_list, my_name = my_val)
```