La Estructura de los Datos en R

Unidad 3: Descripción y Visualización de Datos

Gabriel Sotomayor

2025-09-08

Objetivos de la Sesión de Hoy

- **Comprender** la estructura fundamental de una tabla de datos (observaciones, variables, valores).
- **Reconocer** los tipos de datos básicos en R (`character`, `numeric`, `logical`) y el concepto de coerción.
- **Aprender** un flujo de trabajo básico y reproducible para la exploración inicial de una base de datos.
- **Aplicar** funciones básicas de R para realizar una exploración inicial a un conjunto de datos real.

1. Introducción

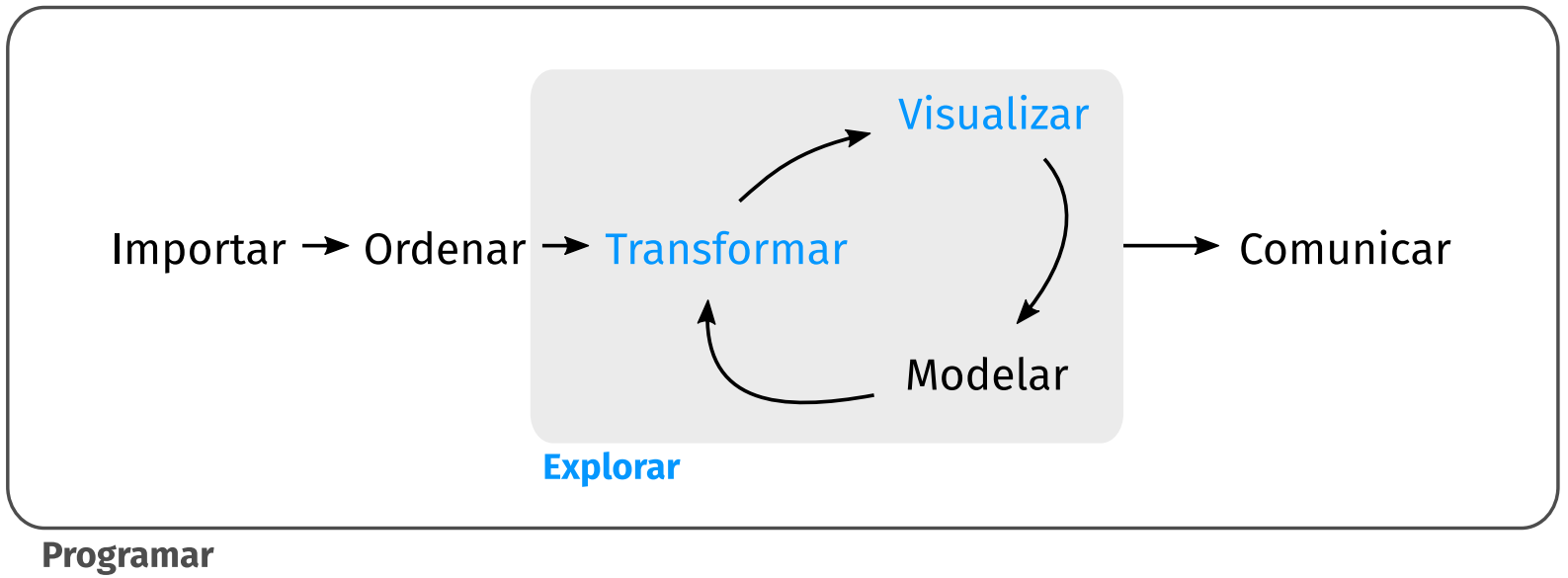
Conectando con la Unidad Anterior

En las clases pasadas, establecimos el camino teórico de la medición:

Conceptualización → Operacionalización → Variables e Indicadores

Hoy, damos el salto a la práctica: **¿Cómo se organizan esas mediciones en el mundo real de los datos?**

Esta unidad se centrará en los primeros y más cruciales pasos del proceso de análisis:



2. Los tipos de datos en R

Tipos de Datos Fundamentales

En R, cada columna de nuestra base de datos tendrá un "tipo" de dato específico. Comprender estos tipos es crucial para evitar errores. Los más comunes son:

Tipo en R	Descripción	Nivel de Medición Típico
`character`	Texto o cadenas de caracteres.	Nominal
`numeric`	Números, ya sean enteros o con decimales.	Intervalo / Razón
`logical`	Valores de Verdadero (`TRUE`) o Falso (`FALSE`).	Nominal (Dicotómica)
`factor`	Variable categórica con niveles predefinidos.	Nominal / Ordinal

Podemos inspeccionar el tipo de una variable con la función `class()`.

El Concepto de Coerción

Una regla fundamental en R es que **todos los elementos de un vector deben ser del mismo tipo**.

Si intentamos combinar diferentes tipos, R forzará o **"coercerá"** los elementos al tipo más flexible para no perder información. La jerarquía de flexibilidad es: `logical` < `numeric` < `character`.

```r
# Creamos un vector con números y un texto
vector_mixto <- c(1, 10, "No sabe")

# ¿Qué tipo de dato tendrá el vector?
class(vector_mixto)
# [1] "character"

# ¿Qué le pasó a los números?
vector_mixto
# [1] "1"  "10"  "No sabe" # ¡Se convirtieron en texto!
```

¡Cuidado! La coerción implícita puede causar errores silenciosos. Si una columna de ingresos tiene un valor de texto ("No responde"), toda la columna puede ser leída como `character`, impidiendo cálculos matemáticos.

Coerción Explícita

Para evitar problemas, la mejor práctica es realizar una **coerción explícita**: nosotros le decimos a R a qué tipo de dato debe convertir una variable.

Usamos la familia de funciones `as.*()`:
- `as.numeric()`: Convierte a número.
- `as.character()`: Convierte a texto.
- `as.factor()`: Convierte a factor.

```r
# Un vector de texto que en realidad son números
edad_texto <- c("25", "42", "37")
class(edad_texto)
# [1] "character"

# No podemos calcular el promedio
mean(edad_texto)
# Error: 'trim' must be numeric

# Lo solucionamos con coerción explícita
edad_numerica <- as.numeric(edad_texto)
class(edad_numerica)
# [1] "numeric"
mean(edad_numerica)
# [1] 34.66667
```

Estructuras de Datos: Los Contenedores de Información

Una vez que entendemos los *tipos* de datos (`numeric`, `character`, etc.), necesitamos saber cómo se *organizan*. En R, los datos se guardan en diferentes "contenedores" o estructuras.

Cada estructura tiene sus propias reglas y usos. Pasaremos de la más simple a la más compleja y útil para nosotros.

1. **Vector**
2. **Lista**
3. **Matriz**
4. **Data Frame** (Nuestra tabla de datos estándar)

1. El Vector: La Variable Individual

El **vector** es la estructura de datos más fundamental en R. Piénsalo como **una sola columna o "variable"** de nuestra base de datos.

- **Regla Clave:** Todos sus elementos deben ser del **mismo tipo** (homogéneo).
- Se crean con la función `c()` (concatenar o combinar).

Ejemplos de vectores:

```r
# Un vector numérico (variable de razón)
edades <- c(25, 30, 22, 45, 38)

# Un vector de texto o "character" (variable nominal)
regiones <- c("Metropolitana", "Valparaíso", "Biobío")

# Un vector lógico (variable nominal dicotómica)
asiste_a_clases <- c(TRUE, TRUE, FALSE, TRUE)
```

La mayoría de las operaciones en R están "vectorizadas", es decir, se aplican a cada elemento del vector a la vez.

2. La Lista: El Contenedor Flexible

Una **lista** es un contenedor que puede guardar cualquier tipo de objeto, incluso de diferentes tipos y tamaños.

- **Regla Clave:** Es **heterogénea**. Puede contener vectores, otros data frames, resultados de modelos, etc.
- Se crean con la función `list()`.

```r
# Creamos una lista con información de un encuestado
encuestado_1 <- list(
  nombre = "Ana",
  edad = 34,
  hijos = c("Pedro", "Javiera"),
  trabaja = TRUE
)

# Podemos acceder a sus elementos con el signo $
encuestado_1$nombre
# [1] "Ana"

encuestado_1$hijos
# [1] "Pedro"  "Javiera"
```

Las listas son extremadamente poderosas para organizar resultados complejos, aunque no las usaremos tanto para guardar datos primarios.

3. La Matriz y el Data Frame

Ambos son estructuras de dos dimensiones (filas y columnas), como una hoja de cálculo, pero con una diferencia crucial.

Matriz (`matrix`)
- **Regla:** Es **homogénea**. Todas las celdas deben contener el mismo tipo de dato (generalmente numérico).
- **Uso:** Principalmente para operaciones matemáticas y álgebra lineal. Menos común para datos de encuesta.

Data Frame (`data.frame`)
- **Regla:** Es **heterogéneo** por columnas. Cada columna (vector) puede tener un tipo de dato diferente, pero todas deben tener la misma longitud.
- **Uso: Es la estructura estándar para almacenar y analizar datos en sociología.** Es nuestra tabla de datos.

Un `data.frame` es, conceptualmente, una **lista de vectores de igual longitud**.

4. El Data Frame: Nuestra Tabla de Datos Estándar

Un `data.frame` organiza nuestros datos de la manera "ordenada" (tidy) que discutimos: cada fila es una observación y cada columna es una variable.

```r
# Creamos vectores individuales
id_sujeto <- c(1, 2, 3)
edad_sujeto <- c(28, 45, 33)
satisfaccion_vida <- c("Alta", "Media", "Alta")

# Los combinamos en un data.frame
mi_encuesta <- data.frame(
  id = id_sujeto,
  edad = edad_sujeto,
  satisfaccion = satisfaccion_vida
)

# Así se ve nuestra base de datos
mi_encuesta
#   id edad satisfaccion
# 1  1   28         Alta
# 2  2   45        Media
# 3  3   33         Alta
```

Al importar datos de un archivo (como haremos en el práctico), R los cargará directamente como un `data.frame` (o un `tibble`, su versión moderna del tidyverse).

La Lógica de una Tabla de Datos

Independientemente de la encuesta o la fuente, los datos cuantitativos se organizan en una estructura tabular con una lógica universal:

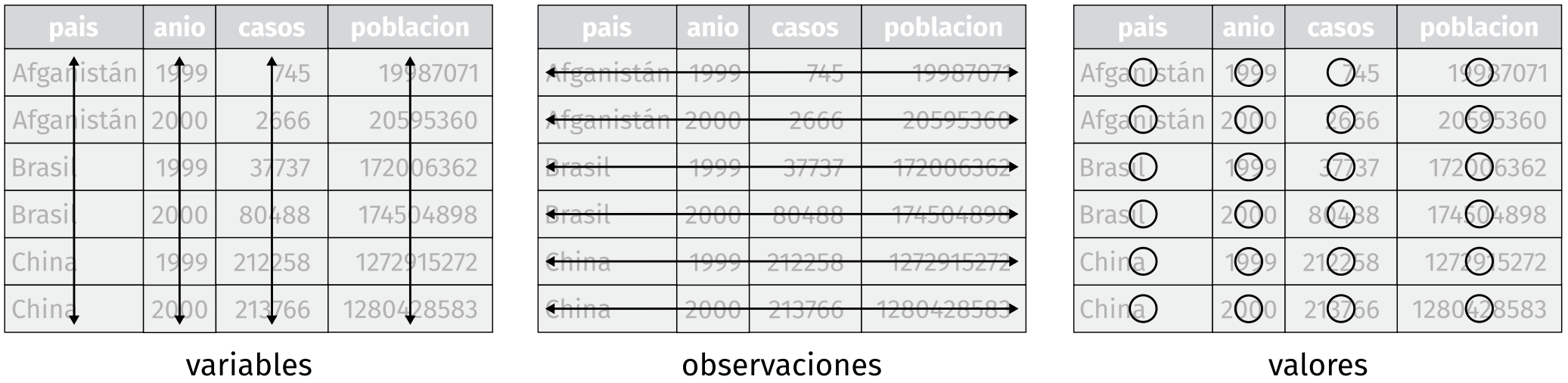
folio	region	sexo	edad	esc	ingreso_hog
101	13	Mujer	45	16	1,500,000
102	5	Hombre	32	12	800,000
102	5	Mujer	30	14	800,000
103	8	Hombre	67	8	450,000

- Cada **Fila** es una **Observación** (o caso). Generalmente, corresponde a nuestra **unidad de análisis** (ej. una persona).
- Cada **Columna** es una **Variable**. Corresponde a un **atributo** o característica que hemos medido.

El Ideal: Datos Ordenados (Tidy Data)

Para que el análisis de datos sea eficiente y libre de errores, buscamos un formato estándar conocido como “**datos ordenados**”. Este formato sigue tres reglas simples:

1. **Cada variable tiene su propia columna.**
2. **Cada observación tiene su propia fila.**
3. **Cada valor tiene su propia celda.**



pais	anio	casos	poblacion
Afganistán	1999	745	19987071
Afganistán	2000	2666	20595360
Brasil	1999	37737	172006362
Brasil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

variables

observaciones

valores

Afortunadamente, la mayoría de las encuestas (como CASEN o EBS) ya vienen en este formato. Nuestro trabajo es mantener este orden.

3. Un Flujo de Trabajo para Explorar Datos

Buenas Prácticas: Script vs. Consola

Antes de analizar, es crucial establecer un flujo de trabajo **reproducible**.

La Consola
- Es la ventana donde R ejecuta los comandos.
- Es interactiva y útil para pruebas rápidas y experimentos.
- **No guarda un registro de tu trabajo**. Lo que escribes en la consola se pierde cuando cierras RStudio.

El Script (.R) o R Markdown (.qmd)
- Son archivos de texto plano donde escribimos y guardamos nuestro código.
- Crean un **registro permanente y reproducible** de cada paso de nuestro análisis.
- **Siempre debemos trabajar en un script o en un R Markdown.**

*Sugerencia: Usar **Proyectos de RStudio (.Rproj)** para mantener todos los archivos de una investigación (datos, scripts, informes) organizados en una sola carpeta.*

Buenas Prácticas para un Código Reproducible

La regla de oro de la reproducibilidad es: **todo debe estar en el script**. Una vez que nos comprometemos con eso, el siguiente paso es asegurarnos de que nuestro código sea claro, robusto y portable.

Para que TÚ (y otros) entiendan tu código:

- **Comentar el código (`#`):**
  - Usa comentarios para explicar el "**porqué**" de tu código, no el "qué". La función `mean()` ya dice que calcula una media; el comentario debe explicar *por qué* necesitas esa media.
- **Nombrar bien las cosas:**
  - Usa nombres de variables y objetos que sean descriptivos y consistentes.
  - **Estilo recomendado:** `snake_case` (minúsculas y guiones bajos).
  - *Mal:* `x <- mean(v1)`
  - *Bien:* `edad_promedio <- mean(edad_encuestados)`
- **Indentar el código:**
  - La indentación (sangría) revela la estructura lógica de tu código, especialmente al usar `%>%`. RStudio lo hace en gran parte de forma automática.

Buenas Prácticas para un Código Reproducible

Para que tu código SIEMPRE funcione:

- **Nunca hacer trabajo manual:**
  - El script debe ser autocontenido. Desde cargar los paquetes y los datos hasta el resultado final, todo debe ser ejecutable sin que tengas que hacer algo con botones que no quede registrado o “arreglar algo a mano en Excel”.
- **Cargar paquetes al inicio:**
  - Toda sesión debe comenzar cargando los paquetes necesarios (ej. `library(tidyverse)`). Así, cualquiera que ejecute tu script sabrá qué necesita instalar.
- **Usar rutas relativas (con Proyectos de RStudio):**
  - Nunca uses rutas absolutas a archivos (ej. `"C:/Users/Gabriel/Desktop/datos.csv"`). Este código fallará en cualquier otro computador.
  - Al usar un Proyecto de RStudio, puedes usar rutas relativas (ej. `"datos/casen2022.csv"`), que funcionarán en cualquier máquina.

La exploración Inicial de los Datos

Cuando recibimos una nueva base de datos, nuestro primer paso es siempre realizar una exploración inicial para entender su estructura y contenido. Estas funciones son nuestro "kit de diagnóstico" básico:

Función	¿Qué hace?
`dim(datos)`	Muestra las **dimensiones** (filas, columnas).
`names(datos)`	Lista los **nombres** de todas las variables.
`head(datos)`	Muestra las primeras 6 **filas**.
`tail(datos)`	Muestra las últimas 6 **filas**.
`str(datos)`	Muestra la **estructura** detallada del objeto.

Interpretando `str()`

La función `str()` (de *structure*) es quizás la más informativa al principio. Nos da un resumen compacto y detallado de cualquier objeto de R, especialmente de las bases de datos (`data.frame`).

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 ...
 $ Species     : Factor w/ 3 levels "setosa",..: 1 1 1 1 1 ...
```

Descripción básica: `summary()` y `table()`

Una vez que entendemos la estructura, pasamos a revisar la calidad del contenido.

- `summary(datos)`:
  - Proporciona un resumen estadístico para **cada variable**.
  - Para variables numéricas: Muestra el mínimo, máximo, media, mediana y cuartiles. **Esencial para detectar valores imposibles** (ej. una edad de -99 o 200).
  - Nos informa cuántos **valores perdidos** (`NA's`) hay en cada variable.
- `table(datos$variable)`:
  - Genera una **tabla de frecuencias** para una variable categórica.
  - Nos permite ver todas las categorías presentes y cuántos casos hay en cada una. Útil para verificar la codificación.

Cierre y Próximos Pasos

Resumen de la sesión de hoy:

- Hemos aprendido que los datos cuantitativos se organizan en una **estructura de filas (observaciones) y columnas (variables)**.
- Entendimos los **tipos de datos básicos en R** y la importancia de controlar la **coerción**.
- Establecimos un **flujo de trabajo inicial** para explorar cualquier nueva base de datos (`dim`, `str`, `summary`, etc.).
- Subrayamos la importancia de las **buenas prácticas** como el uso de scripts para la reproducibilidad.

En el práctico de hoy:

- Aplicarán este flujo de trabajo para realizar una **exploración inicial completa** a la Encuesta CASEN.

Adelanto de la próxima clase:

- Aprenderemos la "gramática de la manipulación de datos" con el paquete `dplyr` para **seleccionar, filtrar y transformar** nuestros datos.