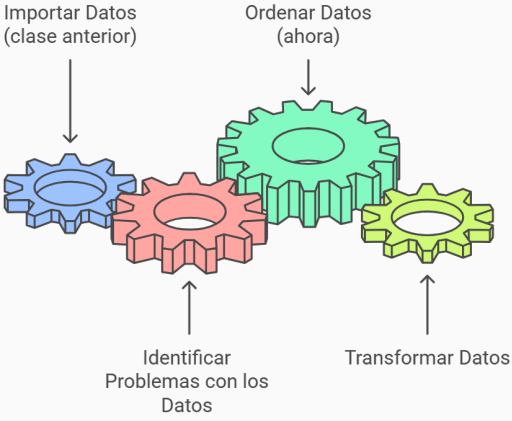
Análisis de datos en fortificación de alimentos a gran escala con R

Tema III: Datos Ordenados.

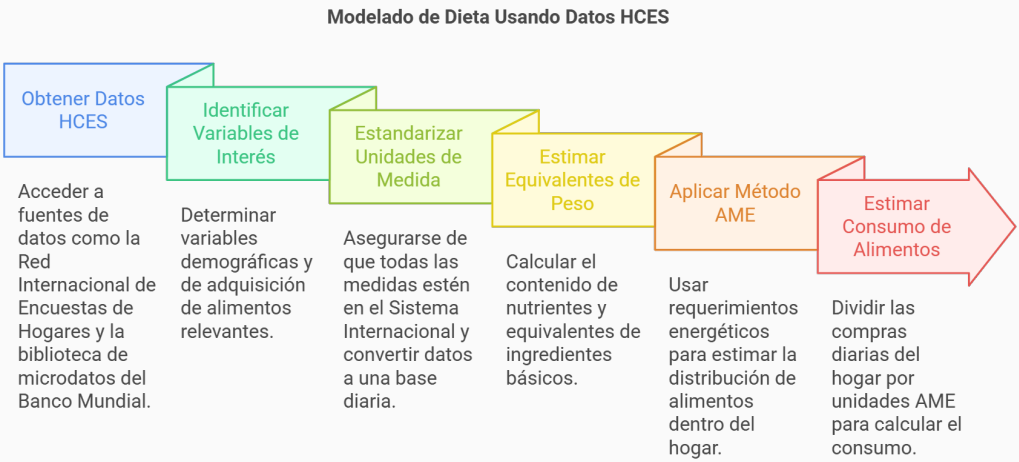
Dr. Maicel Monzón

Clase anterior

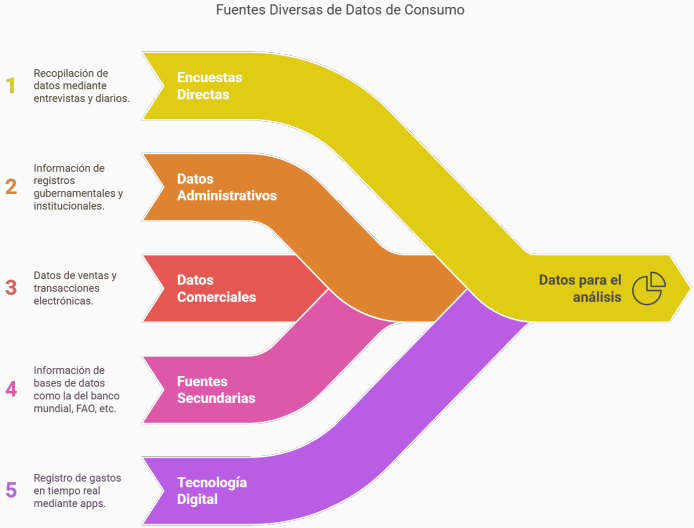
Resumen de lo visto hasta ahora



Para Modelar La Dieta Se Requiere Datos Precisos Y Fáciles De Analizar (ordenados)



Se abordarán métodos para ordenar datos importantos de distintas fuentes.



Sumario

- ¿Qué son los Datos Ordenados?
- ¿Por qué son Importantes?
- Datos Desordenados Comunes
- Funciones Clave de la biblioteca `tidyr`

1. `pivot_longer()`
2. `pivot_wider()`
3. `separate()`
4. `unite()`

"Todas las **familias felices se parecen** unas a otras, pero cada **familia infeliz lo es a su manera**."

León Tolstoy

"Todos los **conjuntos de datos ordenados se parecen** unos a otros, pero cada conjunto de **datos desordenado lo es a su manera**".

Hadley Wickham

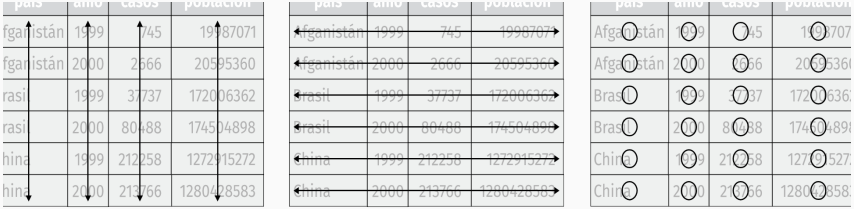
tidydata: Un estándar para organizar datos de manera consistente.

Table 1: conjunto de datos

Col_ID	var 1	var 2	var 3
obs 1	**valor** 1,1	**valor** 1,2	**valor** 1,3
obs ...	**valor** .,1	**valor** .,2	**valor** .,3
obs n	**valor** n,1	**valor** n,2	**valor** n,3

Datos ordenados (Tres Principios)

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



Ejemplo de datos ordenados

Ej. distribución de ingresos y miembros por hogares

```
# A tibble: 3 x 3
  ID_Hogar Ingreso Miembros
     <dbl>   <dbl>    <dbl>
1      101    8500        4
2      102   12000        4
3      102   12000        3
```

Ventajas de datos ordenados

- **Consistencia:** Facilita el aprendizaje.
- **Compatibilidad:** `dplyr`, `gtsummary`, y otros paquetes del tidyverse están diseñados para trabajar con datos ordenados.
- **Eficiencia:** Reduce errores y facilita la manipulación, visualización y análisis.

El enfoque facilita crear nuevas variables con columnas (Eficiencia)

$$pct = Ingreso/Miembros$$

```
hogares_tidy %>%
  mutate(pct=Ingreso/Miembros)
```

```
# A tibble: 3 x 4
  ID_Hogar Ingreso Miembros   pct
     <dbl>   <dbl>    <dbl> <dbl>
1      101    8500        4  2125
2      102   12000        4  3000
3      102   12000        3  4000
```

Es común encontrar datos desodernados

Orígenes del desorden de datos

Formas comunes de datos desordenados

- Problema 1: Una variable distribuida en múltiples columnas.
- Problema 2: Una observación dispersa en múltiples filas.
- Problema 3: Múltiples variables almacenadas en una columna.
- Problema 4: Diferentes tipos de datos almacenados en la misma columna.
- Problema 5: Variables almacenadas tanto en filas como en columnas.

Problema 1: La variable tiempo está distribuida en múltiples columnas

- Se requiere convertir de **formato ancho** a **formato largo**

Datos desordenados

1: Variable distribuida en múltiples columnas

Alimento	Enero	Febrero	Marzo
Arroz	150	140	160
Pollo	100	90	110
Verduras	200	180	210

Formato ancho

Pivotar

A lo largo

Formato Largo

Datos ordenados

Alimento	Mes	Gramos
Arroz	Enero	150
Arroz	Febrero	140
Arroz	Marzo	160
Pollo	Enero	100
Pollo	Febrero	90
Pollo	Marzo	110
Verduras	Enero	200
Verduras	Febrero	180
Verduras	Marzo	210

Problema 2: La variable nutriente está dispersa en múltiples filas

- Se requiere convertir de **formato largo** a **formato ancho**

Datos desordenados

2: Una observación dispersa en múltiples filas.

Formato Largo

ID	nutriente	cantidad
101	Proteína	70
101	Carbohidratos	250
102	Proteína	60
102	Carbohidratos	300

Datos ordenados

ID	Proteína	Carbohidratos
101	70	250
102	60	300

Formato ancho

Problema 3: Múltiples variables almacenadas en una misma columna.

- Alimento y comida almacenadas en la variable **Alimento_Comida** separada por un guión requiere separar la variable.

Datos desordenados

3: Múltiples variables almacenadas en una columna.

Alimento_Comida	Gramos
Leche_Desayuno	200
Pan_Cena	50
Carne_Almuero	150

separar

Datos ordenados

Alimento	Comida	Gramos
Leche	Desayuno	200
Pan	Cena	50
Carne	Almuero	150

Problema 4: Diferentes tipos de datos almacenados en la misma columna.

- Se requiere **Unir** el día, mes y el año para construir la fecha.

Datos desordenados

4: Diferentes tipos de datos almacenados en la misma columna.

ID	año	mes	dia
1	2,023	10	26
2	2,023	11	15
3	2,024	1	1

unir

Datos ordenados

```
# A tibble: 3 × 2
     ID fecha
  <dbl> <chr>
1     1 2023-10-26
2     2 2023-11-15
3     3 2024-1-1
```

Funciones principales tidyr(Una biblioteca para ordenar datos)

Funciones principales

- pivot_longer
- pivot_wider
- separate
- unite

tidyr::pivot_longer()

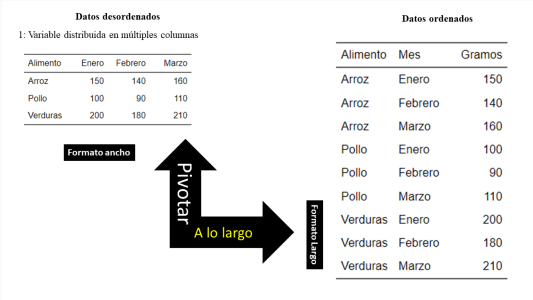
- **Función:** Convierte datos “anchos” a “largos”.
- **Uso:** Cuando los nombres de las columnas son valores, no variables.

Argumentos principales de `pivot_longer()`

- `cols`: Indica qué columnas **quieres "alargar"** (convertir de formato ancho a largo)
- `names_to`: Define el nombre de la nueva columna que almacenará los nombres originales de las columnas que especificaste en **cols**
- `values_to`: Define el nombre de la nueva columna que almacenará los valores de las columnas especificadas en cols.

Ejemplo: Convertiendo de ancho a largo

```
ingesta_tidy <- ingesta_mensual %>%
  pivot_longer(cols = Enero:Marzo, # columnas original (DF viejo)
               names_to = "Mes", # nueva columna de nombre (DF Nuevo)
               values_to = "Gramos"# nueva columna de valores (DF Nuevo)
               )
```



tidyr::pivot_wider()

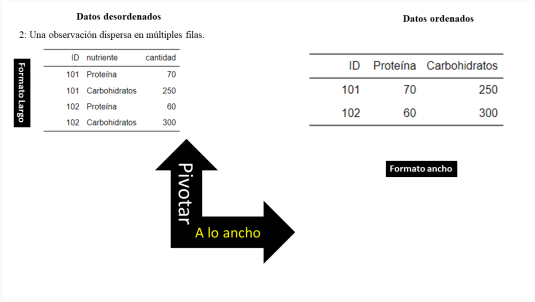
- **Función:** Convierte datos “largos” a “anchos”.
- **Uso:** Observación dispersa en filas.

Argumentos Clave de `pivot_wider()`

- `names_from`: Columna para **nombres** del nuevo DF
- `values_from`: Columna para **valores** del nuevo DF.

Ejemplo: Convertiendo de largo a ancho

```
ingesta_ordenada <- ingesta_desordenada_id %>%
  pivot_wider(names_from = nutriente, # nombres en el nuevo DF
              values_from = cantidad #  valores en el nuevo DF
              )
```



tidyr::separate()

- **Función:** Divide una columna en múltiples columnas
- **Uso:** Cuando una columna contiene múltiples variables combinadas

Argumentos Clave de `separate()`

- `cols` : La columna que se va a separar
- `into` : Vector de nombres para las nuevas columnas
- `sep`: El carácter o la posición donde se va a separar la columna. Ej “_”

Ejemplo: Separando columnas

```
consumo_tidy <- consumo %>%
  separate(Alimento_Comida,
           into = c("Alimento", "Comida"),
           sep = "_")
```

Datos desordenados

3: Múltiples variables almacenadas en una columna.

Alimento_Comida	Gramos
Leche_Desayuno	200
Pan_Cena	50
Carne_Almuero	150

separar

Datos ordenados

Alimento	Comida	Gramos
Leche	Desayuno	200
Pan	Cena	50
Carne	Almuero	150

tidyr::unite()

- **Función:** Combina múltiples columnas en una sola columna
- **Uso:** Cuando los componentes de una sola variable están dispersos en múltiples columnas

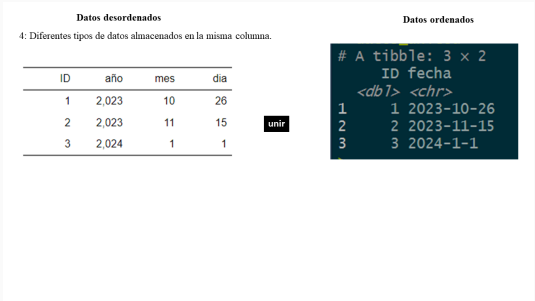
Argumentos Clave de `unite()`

- `cols` : El nombre de la nueva columna combinada
- `...` : Las columnas que se van a unir
- `sep`: El carácter separado

-`remove`: Eliminar las columnas de entrada

Ejemplo: uniendo columnas

```r
dates_united <- date_components %>%
  unite(
    fecha,      # Nombre de la nueva columna
    año, mes, dia, # Columnas a combinar
    sep = "-"     # Carácter separador
  )
```



Conclusiones

"Hemos estudiado cómo transformar datos caóticos en **datos ordenados**, utilizando funciones de *tidyr* como `pivot_longer()`, `pivot_wider()`, `unite()` y `separate()`. Al dominar estos métodos, convertimos la complejidad en claridad, permitiendo **transformaciones más eficientes** y **análisis de datos más precisos**. ¡Organicemos datos y construyamos conocimiento!"

!Muchas Gracias!