Introducción a la ciencia de datos

Capítulo 3: Almacenamiento y lectura de datos

Mg. J. Eduardo Gamboa U.

2025-04-01

Iniciando un proyecto en ciencia de datos

- Primer paso: definir los objetivos = f(problema)
- ¿Qué? ¿Por qué? ¿Cómo?
- ¿Cuáles son las expectativas de las partes involucradas (alta dirección, desarrolladores, analistas, usuarios del sistema, clientes)?
- ¿Cómo el proyecto va a cambiar el negocio?
- ¿Cómo se usarán los resultados?
- Cronograma del proyecto con acuerdos de entregables (fechas, responsables, porcentaje de avance, asignación de recursos, etc)

Almacenamiento en Big Data

Sistemas de almacenamiento

01	**Repositorios**	• Base de datos • Data mart • Data warehouse • Data lake
02	**Tecnologías**	• Apache Hadoop • MongoDB • Cassandra • Apache HBase

Repositorios

1. **Data lake**: Es un medio de almacenamiento de data estructurada , semiestructurada y no estructurada (en su formato original). Facilita la organización y gestión de grandes volúmenes de datos provenientes de diversas fuentes, como **CRM**, **ERP**, Redes sociales, **IoT** y/o sensores.
2. **Data warehouse (DWH)**: Es un sistema centralizado de almacenamiento que integra datos de múltiples fuentes, generalmente estructurados. Aunque su enfoque principal es el manejo de datos estructurados, también puede procesar datos no estructurados (utilizando **ETL**).
3. **Data mart**: Es un almacén de datosdiseñado para un área específica dentro de una organización. Usualmente, es un subconjunto de un Data Warehouse, optimizado para consultas y análisis en un dominio particular.
4. **Database (base de datos)**: Es una colección organizada de datos estructurados en tablas (bases de datos relacionales **SQL**) o en formatos flexibles y no estructurados (noSQL)

DATA WAREHOUSE

VS

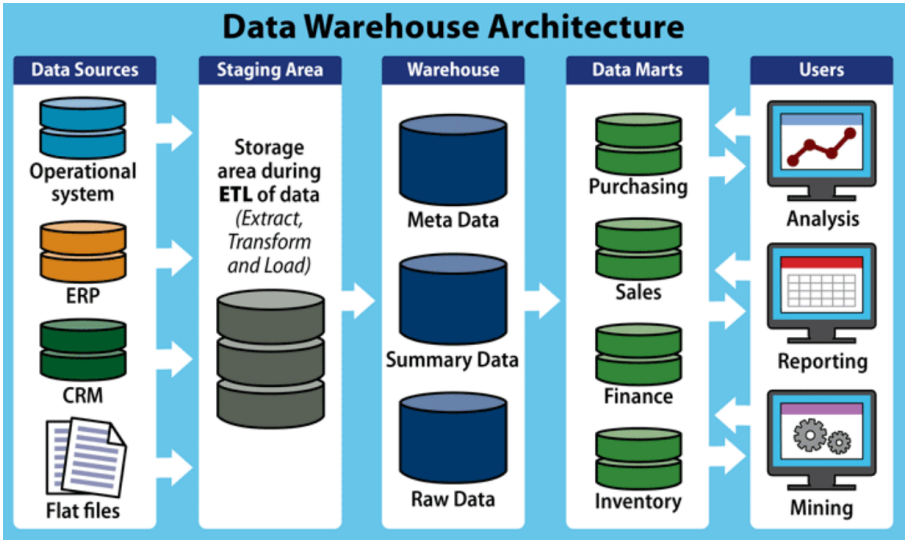
DATA LAKE

Data is processed and organized into a single schema before being put into the warehouse

The analysis is done on the cleansed data in the warehouse

Raw and unstructured data goes into a data lake

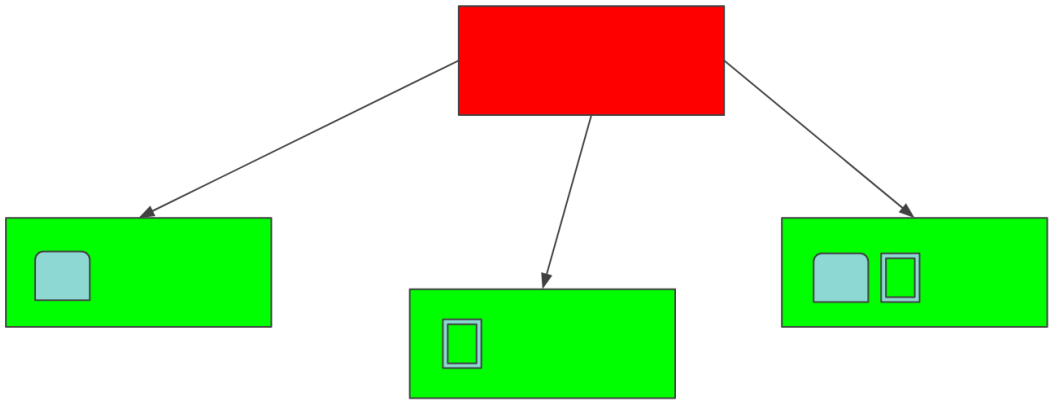
Data is selected and organized as and when needed


Data Warehouse Architecture
Data Sources
Operational system
ERP
CRM
Flat files
Staging Area
Storage area during ETL of data (Extract, Transform and Load)
Warehouse
Meta Data
Summary Data
Raw Data
Data Marts
Purchasing
Sales
Finance
Inventory
Users
Analysis
Reporting
Mining

Tecnologías

Apache Hadoop

- Es un entorno de trabajo (ecosistema) de software de código abierto que permite el almacenamiento de datos en clusters de servidores.
- Un cluster está conformado por un nodo o máquina maestra (primaria) y múltiples nodos secundarios (ex - esclavos).
- Hadoop fue creado por Doug Cutting y Mike Cafarella en 2006, inspirado por los trabajos de Google sobre el procesamiento y almacenamiento de datos a gran escala.
- Fue lanzado oficialmente por la Apache Software Foundation en 2011.



- Actualmente es gestionado y mantenido por la Apache Software Foundation (ASF).
- Licenciado bajo la Licencia Apache 2.0, permitiendo así su uso, modificación y distribución.
- Hadoop es altamente escalable, lo que permite añadir o quitar nodos según sea necesario.
- También ofrece tolerancia a fallos replicando los datos en múltiples nodos del cluster.

- Almacena todo tipo de datos: estructurado y no estructurado, y metadatos
- Sus principales componentes son **HDFS** y MapReduce.
- Diversas empresas usan servicios y soluciones comerciales basadas en Hadoop, ofrecidos por Cloudera, IBM, Amazon Web Services, Microsoft, Google, etc.
- Empresas que usan Hadoop: Adobe, Alibaba, EBay, Facebook, Hulu, IBM, LinkedIn, MercadoLibre, The New York Times, Spotify, Telefónica, Twitter, Yahoo, etc. Más empresas que usan Hadoop aquí.
- No es eficiente para tareas iterativas y/o interactivas (a menos que se use en conjunto con Apache Spark)

mongoDB

- Viene del término inglés humongous
- Fue lanzado en el año 2009 por 10gen (ahora MongoDB Inc.)
- Maneja sistema de base de datos **no SQL** (no relacional)
- Almacena los datos en una extensión del formato **JSON**.
- Es recomendable usarlo cuando nuestros datos están almacenados en la nube, varían mucho (en tiempo real) y varios usuarios acceden a ella.
- Empresas que usan mongoDB: The Weather Channel, Adobe, Nokia, eBay, Google, Facebook, PayPal, Telefónica. Más empresas aquí.

Lectura de datos estructurados

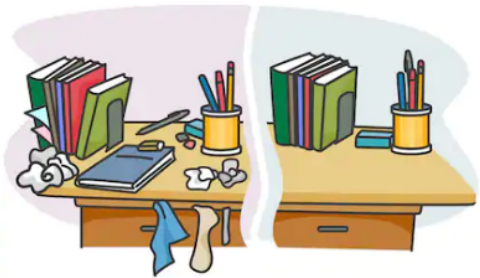
Archivos de datos

Los datos pueden estar contenidos en archivos de distintos formatos, entre los que se tiene los más comunes:

1. Archivo de texto plano: su contenido es netamente textual, sin ningún formato. Lleva la extensión .txt y puede ser creado y/o editado en Bloc de notas.
2. Archivo de valores separados por comas: permite representar datos en formato tabular, en el que las columnas son separadas por comas (o puntos y comas, dependiendo del país y/o la configuración de la computadora). Lleva la extensión .csv y puede ser creado y/o editado en programas como Bloc de notas o Excel.
3. Archivo de hoja de cálculo: permite almacenar los datos en celdas, dispuestas en filas y columnas. La extensión común es .xlsx o .xls

Antes de iniciar con la lectura de datos...

...debemos entender y ordenar nuestro espacio de trabajo



¿Cómo RStudio accede al archivo que contiene nuestro conjunto de datos? ¿Dónde "vive" nuestro código de análisis de datos?

Ejecute el código:

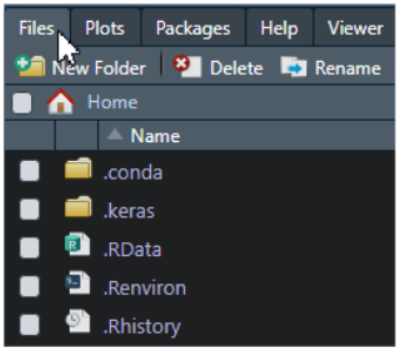
```
getwd()
```

Por defecto, R ejecutará todos nuestros análisis en esta carpeta. ¿Cómo cambiamos esa ruta?

1. Usar la función `setwd(...)`
2. Usar proyecto(s): opción reproducible

Entorno de trabajo

¿Quiénes son los "vecinos" de nuestro código de análisis de datos?
Puedes verlo en la pestaña Files del panel inferior derecho.
Se listan todos los archivos que conforman la carpeta que obtuvimos con la función `getwd()` o la que fijamos con `setwd(...)`.



Reproducibilidad

La reproducibilidad es importante. Si tú ejecutas un código y se lo compartes a otra persona, debe obtener el mismo resultado, sin necesidad de realizar cambios en el código.

¿Qué sucede si un código comienza con `setwd("C:/Users/Usuario 1")`?

Proyectos en RStudio

Permite manejar el entorno de trabajo de manera ordenada. Todos los archivos quedan empaquetados en una carpeta de trabajo.

¿Cómo crear un proyecto en RStudio?

1. Crear una carpeta de trabajo, la cual contiene o contendrá todos los archivos necesarios para el análisis.
2. Click en File $\rightarrow$ New Project o hacer click en los íconos que ejecutan esa misma acción
3. Elegir "Existing Directory" y luego en "Browse" buscar la carpeta de trabajo

Lectura de datos

Utilizaremos distintas funciones, como por ejemplo `read.table`, `read_csv2` o `read_xlsx`, pero antes de ello necesitaremos instalar los paquetes `readr` y `readxl`.

```
install.packages("readr")
install.packages("readxl")
```

No son los únicos paquetes que existen para lectura de datos.

read.table

```
datos <- read.table(file, header, sep, dec, col.names, row.names,
                    na.strings, nrows, skip)
```

- `file`: Argumento character obligatorio. Corresponde al archivo de datos a ser leído. Va entre comillas. Incluir su extensión (txt, csv), por ejemplos `"datos336.txt"`
- `header`: Argumento logical opcional. `TRUE` si la primera línea contiene los nombres de las variables. Por defecto se asume que es `FALSE`, a menos que el número de campos en la primera fila sea uno menos que el número de columnas.
- `sep`: Argumento character opcional. Indica el separador entre columnas. Va entre comillas, por defecto asume que es `sep = " "` (al menos un espacio en blanco).
- `dec`: Argumento character opcional. Indica el separador de la parte entera y decimal en los números, por defecto asume `dec = "."`

```
datos <- read.table(file, header, sep, dec, col.names, row.names,
                    na.strings, nrows, skip)
```

- `col.names`: Argumento de tipo vector character, opcional. Indica el nombre de las columnas.
- `row.names`: Argumento de tipo múltiple, opcional. Puede ser un vector character que indica el nombre de las filas. También se puede indicar el número de columna contiene estos nombres.
- `na.strings`: Argumento character opcional. Indica cómo se declaran los valores perdidos (Not Available).
- `nrows`: Argumento integer opcional. Indica el número máximo de filas a leer. Por defecto es -1, de modo que dicho valor es ignorado (y así lee el archivo completo).
- `skip`: Argumento integer opcional. Indica el número de filas iniciales del archivo que no serán leídas. Por defecto es 0.

Ejemplo 1

El archivo `Cap3 - INEI - 001.txt` contiene datos sobre la población de 0 a 5 años estimada por el INEI para el año 2021 en el distrito de Bellavista, mientras que el archivo `Cap3 - INEI - Diccionario.xlsx` proporciona el diccionario de datos. Ejecutar los siguientes códigos para leer el archivo. ¿Realizan la misma acción? ¿En qué se diferencian?

Pista: Usar la función `identical`

```
datos1a = read.table(file = "Cap3 - INEI - 001.txt", header = TRUE)
datos1b = read.table("Cap3 - INEI - 001.txt", TRUE)
datos1c = read.table(header = TRUE, file = "Cap3 - INEI - 001.txt")
```

read_tsv

Una vez instalado el paquete `readr`, cargarlo:

```
library(readr)
```

```
datos <- read_tsv(file, col_names, na, skip, n_max)
```

- `file`: Argumento character obligatorio. Corresponde al archivo de datos a ser leído. Va entre comillas. Incluir su extensión (txt), por ejemplos `"datos123.txt"`
- `col_names`: Argumento opcional de múltiple tipo. Puede ser logical (`TRUE` o `FALSE`) si indica la presencia o ausencia de nombres de columnas (como `header` en `read.table`), o puede ser un vector indicando los nombres de las columnas. Por defecto toma el valor `TRUE`
- `na`: Argumento de tipo vector character opcional. Indica cómo se declaran los valores perdidos. Por defecto es `c("", "NA")`.

- `skip`: Argumento numérico opcional. Indica el número de filas iniciales del archivo que no serán leídas. Por defecto es 0.
- `n_max`: Argumento numérico opcional. Indica el número máximo de filas a leer. Por defecto es infinito (`Inf`).

Ejercicio 1

Volver a leer el archivo `Cap3 - INEI - 001.txt` utilizando la función `read_tsv`. Tener en cuenta que los argumentos pueden variar en nombre y no son necesariamente los mismos, además de que no son los únicos disponibles. El único argumento obligatorio es `file`.

read.csv

```
datos <- read.csv(file, header, sep, dec, col.names, row.names,
                  na.strings, nrows, skip)
```

- `file`: Argumento character obligatorio. Corresponde al archivo de datos a ser leído. Va entre comillas. Incluir su extensión (txt, csv), por ejemplos `"datos159.csv"`
- `header`: Argumento logical opcional. `TRUE` si la primera línea contiene los nombres de las variables. Por defecto se asume que es `TRUE`.
- `sep`: Argumento character opcional. Indica el separador entre columnas. Va entre comillas, por defecto asume que es `sep = ","` (coma).
- `dec`: Argumento character opcional. Indica el separador de la parte entera y decimal en los números, por defecto asume `dec = "."`

```
datos <- read.csv(file, header, sep, dec, col.names, row.names,
                  na.strings, nrows, skip)
```

- `col.names`: Argumento de tipo vector, opcional. Indica el nombre de las columnas.
- `row.names`: Argumento de tipo vector, opcional. Indica el nombre de las filas. También se puede indicar que una columna contiene estos nombres.
- `na.strings`: Argumento character opcional. Indica cómo se declaran los valores perdidos (Not Available).
- `nrows`: Argumento integer opcional. Indica el número máximo de filas a leer.
- `skip`: Argumento integer opcional. Indica el número de filas iniciales del archivo que no serán leídas.

Ejemplo 2

Ejecutar los siguientes códigos para leer el archivo `Cap3 - INEI - 002.csv`. ¿Realizan la misma acción? ¿En qué se diferencian? ¿Cuál prefieres utilizar?

```
datos2a = read.table("Cap3 - INEI - 002.csv", TRUE, sep = ",")
datos2b = read.csv("Cap3 - INEI - 002.csv")
```

read_csv

```
datos <- read_csv(file, col_names, na, skip, n_max)
```

- `file`: Argumento character obligatorio. Corresponde al archivo de datos a ser leído. Va entre comillas. Incluir su extensión (csv), por ejemplos `"datos927.csv"`
- `col_names`: Argumento opcional de múltiple tipo. Puede ser logical (`TRUE` o `FALSE`) si indica la presencia o ausencia de nombres de columnas (como `header` en `read.table`), o puede ser un vector indicando los nombres de las columnas. Por defecto toma el valor `TRUE`
- `na`: Argumento de tipo vector character opcional. Indica cómo se declaran los valores perdidos. Por defecto es `c("", "NA")`.
- `skip`: Argumento numérico opcional. Indica el número de filas iniciales del archivo que no serán leídas. Por defecto es 0.
- `n_max`: Argumento numérico opcional. Indica el número máximo de filas a leer. Por defecto es infinito (`Inf`).

Ejercicio 2

Volver a leer el archivo `Cap3 - INEI - 002.csv` utilizando la función `read_csv`. Tener en cuenta que los argumentos pueden variar en nombre y no son necesariamente los mismos, además de que no son los únicos disponibles. El único argumento obligatorio es `file`.

read.csv2

```
datos <- read.csv2(file, header, sep, dec, col.names, row.names,
                   na.strings, nrows, skip)
```

- `file`: Argumento character obligatorio. Corresponde al archivo de datos a ser leído. Va entre comillas. Incluir su extensión (txt, csv), por ejemplos `"datos159.csv"`
- `header`: Argumento logical opcional. `TRUE` si la primera línea contiene los nombres de las variables. Por defecto se asume que es `TRUE`.
- `sep`: Argumento character opcional. Indica el separador entre columnas. Va entre comillas, por defecto asume que es `sep = ";"` (coma).
- `dec`: Argumento character opcional. Indica el separador de la parte entera y decimal en los números, por defecto asume `dec = ","`

```
datos <- read.csv2(file, header, sep, dec, col.names, row.names,
                   na.strings, nrows, skip)
```

- `col.names`: Argumento de tipo vector, opcional. Indica el nombre de las columnas.
- `row.names`: Argumento de tipo vector, opcional. Indica el nombre de las filas. También se puede indicar que una columna contiene estos nombres.
- `na.strings`: Argumento character opcional. Indica cómo se declaran los valores perdidos (Not Available).
- `nrows`: Argumento integer opcional. Indica el número máximo de filas a leer.
- `skip`: Argumento integer opcional. Indica el número de filas iniciales del archivo que no serán leídas.

Ejemplo 3

Ejecutar los siguientes códigos para leer el archivo `Cap3 - INEI - 003.csv`. ¿Realizan la misma acción? ¿En qué se diferencian? ¿Cuál prefieres utilizar?

```
datos3a = read.table("Cap3 - INEI - 003.csv", TRUE, sep = ";")
datos3b = read.csv("Cap3 - INEI - 003.csv", sep = ";")
datos3c = read.csv2("Cap3 - INEI - 003.csv")
```

read_csv2

```
datos <- read_csv2(file, col_names, na, skip, n_max)
```

- `file`: Argumento character obligatorio. Corresponde al archivo de datos a ser leído. Va entre comillas. Incluir su extensión (csv), por ejemplos `"datos927.csv"`
- `col_names`: Argumento opcional de múltiple tipo. Puede ser logical (`TRUE` o `FALSE`) si indica la presencia o ausencia de nombres de columnas (como `header` en `read.table`), o puede ser un vector indicando los nombres de las columnas. Por defecto toma el valor `TRUE`
- `na`: Argumento de tipo vector character opcional. Indica cómo se declaran los valores perdidos. Por defecto es `c("", "NA")`.
- `skip`: Argumento numérico opcional. Indica el número de filas iniciales del archivo que no serán leídas. Por defecto es 0.
- `n_max`: Argumento numérico opcional. Indica el número máximo de filas a leer. Por defecto es infinito (`Inf`).

Ejercicio 3

Volver a leer el archivo `Cap3 - INEI - 003.csv` utilizando la función `read_csv2`. Tener en cuenta que los argumentos pueden variar en nombre y no son necesariamente los mismos, además de que no son los únicos disponibles. El único argumento obligatorio es `file`.

Investigar acerca de las otras funciones disponibles en los paquetes `utils` y `readr`, y practicar (leer los archivos ya utilizados).

Funciones del paquete `utils`: `read.delim` y `read.delim2`.

Función del paquete `readr`: `read_delim`.

read_xls, read_xlsx y read_excel

Cargar el paquete `readxl`

```
library(readxl)
```

```
datos <- read_excel(path, sheet, range, col_names, na, skip, n_max)
```

- `path`: Argumento character obligatorio. Corresponde al archivo de datos de extensión .xls o .xlsx a ser leído. Va entre comillas. Incluir su extensión, por ejemplo `datos333.xlsx`.
- `sheet`: Argumento de tipo variable. Puede ser de tipo character indicando el nombre de la hoja a ser leída, o de tipo entero indicando la posición de la hoja. Es ignorado si se indica el nombre de hoja en `range`.
- `range`: Argumento character opcional. Indica el rango de celdas a ser leído. Por ejemplo `"B3:D87"` o `"Budget!B2:G14"`, donde `Budget` es el nombre de la hoja. Este argumento es más importante que `sheet`, `skip` y `n_max`.

```
datos <- read_excel(path, sheet, range, col_names, na, skip, n_max)
```

- `col_names`: Argumento opcional de múltiple tipo. Puede ser logical (`TRUE` o `FALSE`) si indica la presencia o ausencia de nombres de columnas (como `header` en `read.table`), o puede ser un vector indicando los nombres de las columnas. Por defecto toma el valor `TRUE`
- `na`: Argumento de tipo vector character opcional. Indica cómo se declaran los valores perdidos. Por defecto una celda en blanco es un valor perdido.
- `skip`: Argumento numérico opcional. Indica el número de filas iniciales del archivo que no serán leídas. Si hay líneas completas vacías al inicio del archivo son automáticamente ignoradas. Por defecto es 0. Es ignorado si se usa el argumento `range`.
- `n_max`: Argumento numérico opcional. Indica el número máximo de filas a leer. Si hay líneas completas vacías al final del archivo son automáticamente ignoradas. Por defecto es infinito (`Inf`). Es ignorado si se usa el argumento `range`.

Ejemplo 4

Ejecutar los siguientes códigos para leer los archivos `Cap3 - INEI - 004.xlsx` y `Cap3 - INEI - 005.xls`. ¿Realizan la misma acción? ¿En qué se diferencian? ¿Cuál prefieres utilizar?

```
datos4a = read_excel("Cap3 - INEI - 004.xlsx")
datos4b = read_xlsx("Cap3 - INEI - 004.xlsx")
datos4c = read_excel("Cap3 - INEI - 005.xls")
datos4d = read_xls("Cap3 - INEI - 005.xls")
```

Uso de la función `head`

```
head(datos4a)
```

```
# A tibble: 6 x 8
  ubigeo_reniec ubigeo_inei Departamento Provincia Distrito   Edad_Anio Sexo
          <dbl> <chr>       <chr>        <chr>     <chr>          <dbl> <chr>
1        240102 070102      CALLAO       CALLAO    BELLAVISTA         5 F
2        240102 070102      CALLAO       CALLAO    BELLAVISTA         5 M
3        240102 070102      CALLAO       CALLAO    BELLAVISTA         4 F
4        240102 070102      CALLAO       CALLAO    BELLAVISTA         4 M
5        240102 070102      CALLAO       CALLAO    BELLAVISTA         3 F
6        240102 070102      CALLAO       CALLAO    BELLAVISTA         3 M
# i 1 more variable: Cantidad <dbl>
```

```r
head(datos4a, n = 3)
```

```
# A tibble: 3 x 8
  ubigeo_reniec ubigeo_inei Departamento Provincia Distrito   Edad_Anio Sexo
          <dbl> <chr>       <chr>        <chr>     <chr>          <dbl> <chr>
1        240102 070102      CALLAO       CALLAO    BELLAVISTA         5 F
2        240102 070102      CALLAO       CALLAO    BELLAVISTA         5 M
3        240102 070102      CALLAO       CALLAO    BELLAVISTA         4 F
# i 1 more variable: Cantidad <dbl>
```

Uso de la función `tail`

```
tail(datos4a)
```

```
# A tibble: 6 x 8
  ubigeo_reniec ubigeo_inei Departamento Provincia Distrito   Edad_Anio Sexo
          <dbl> <chr>       <chr>        <chr>     <chr>          <dbl> <chr>
1        240102 070102      CALLAO       CALLAO    BELLAVISTA         2 F
2        240102 070102      CALLAO       CALLAO    BELLAVISTA         2 M
3        240102 070102      CALLAO       CALLAO    BELLAVISTA         1 F
4        240102 070102      CALLAO       CALLAO    BELLAVISTA         1 M
5        240102 070102      CALLAO       CALLAO    BELLAVISTA         0 F
6        240102 070102      CALLAO       CALLAO    BELLAVISTA         0 M
# i 1 more variable: Cantidad <dbl>
```

```r
tail(datos4a, n = 8)
```

```
# A tibble: 8 x 8
  ubigeo_reniec ubigeo_inei Departamento Provincia Distrito   Edad_Anio Sexo
          <dbl> <chr>       <chr>        <chr>     <chr>          <dbl> <chr>
1        240102 070102      CALLAO       CALLAO    BELLAVISTA         3 F
2        240102 070102      CALLAO       CALLAO    BELLAVISTA         3 M
3        240102 070102      CALLAO       CALLAO    BELLAVISTA         2 F
4        240102 070102      CALLAO       CALLAO    BELLAVISTA         2 M
5        240102 070102      CALLAO       CALLAO    BELLAVISTA         1 F
6        240102 070102      CALLAO       CALLAO    BELLAVISTA         1 M
7        240102 070102      CALLAO       CALLAO    BELLAVISTA         0 F
8        240102 070102      CALLAO       CALLAO    BELLAVISTA         0 M
# i 1 more variable: Cantidad <dbl>
```