EDA Salud Mental

Table of contents

0.1 Librerías

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   4.0.0     v tibble    3.2.1
```

```
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ---------------------------------------- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```r
library(readxl)
library(visdat)
library(GGally)
library(patchwork)
library(writexl)
library(lubridate)
library(stringr)
library(uuid)

# Opciones de gráficos para acelerar el render a PDF
knitr::opts_chunk$set(
  dev = "png",
  dpi = 200,
  fig.width = 11,
  fig.height = 6.5,
  out.width = "100%"
)
```

0.2 Carga de datos

Leemos el Excel original. Si aparecen warnings en la carga, se muestran justo encima de esta línea en la salida y los comentamos a continuación.

```r
SaludMental <- read_excel("../SaludMental.xls")
```

```
Warning: Expecting logical in CX1831 / R1831C102: got '0UT94ZZ'

Warning: Expecting logical in CY1831 / R1831C103: got '0UT24ZZ'

Warning: Expecting logical in CZ1831 / R1831C104: got '0UT74ZZ'

Warning: Expecting logical in AL5125 / R5125C38: got 'GZB4ZZZ'
```

```
Warning: Expecting logical in AL5554 / R5554C38: got 'GZB4ZZZ'

Warning: Expecting logical in AM5554 / R5554C39: got 'GZB4ZZZ'

Warning: Expecting logical in AN5554 / R5554C40: got 'GZB4ZZZ'

Warning: Expecting logical in AO5554 / R5554C41: got 'GZB4ZZZ'

Warning: Expecting logical in AP5554 / R5554C42: got 'GZB4ZZZ'

Warning: Expecting logical in AQ5554 / R5554C43: got 'GZB4ZZZ'

Warning: Expecting logical in AR5554 / R5554C44: got 'GZB4ZZZ'

Warning: Expecting logical in AS5554 / R5554C45: got 'GZB4ZZZ'

Warning: Expecting logical in AT5554 / R5554C46: got 'GZB4ZZZ'

Warning: Expecting logical in AL5875 / R5875C38: got 'GZB4ZZZ'

Warning: Expecting logical in AL6786 / R6786C38: got 'GZB4ZZZ'

Warning: Expecting logical in AM6786 / R6786C39: got 'GZB4ZZZ'

Warning: Expecting logical in AN6786 / R6786C40: got 'GZB4ZZZ'

Warning: Expecting logical in AO6786 / R6786C41: got 'GZB4ZZZ'

Warning: Expecting logical in AP6786 / R6786C42: got 'GZB4ZZZ'

Warning: Expecting logical in AQ6786 / R6786C43: got 'GZB4ZZZ'

Warning: Expecting logical in AR6786 / R6786C44: got 'GZB4ZZZ'

Warning: Expecting logical in AL9025 / R9025C38: got 'GZB0ZZZ'

Warning: Expecting logical in AM9025 / R9025C39: got 'GZB0ZZZ'
```

```
Warning: Expecting logical in AN9025 / R9025C40: got 'GZB0ZZZ'

Warning: Expecting logical in AO9025 / R9025C41: got 'GZB0ZZZ'

Warning: Expecting logical in AL9452 / R9452C38: got 'GZB4ZZZ'

Warning: Expecting logical in AM9452 / R9452C39: got 'GZB4ZZZ'

Warning: Expecting logical in AL10005 / R10005C38: got 'GZB4ZZZ'

Warning: Expecting logical in AL10637 / R10637C38: got 'GZB4ZZZ'

Warning: Expecting logical in AM10637 / R10637C39: got 'GZB4ZZZ'

Warning: Expecting logical in CX10722 / R10722C102: got 'B2111ZZ'

Warning: Expecting logical in AL11057 / R11057C38: got 'GZB4ZZZ'

Warning: Expecting logical in AL12036 / R12036C38: got 'GZB4ZZZ'

Warning: Expecting logical in AL14754 / R14754C38: got 'GZ50ZZZ'

Warning: Expecting logical in CX14908 / R14908C102: got '0FC44ZZ'
```

0.3 Objetivos del plan de limpieza

1. Preservar el dataset original (`SaludMental`) y construir una copia (`SaludMental_limpio`).
2. Eliminar columnas sin información (100% NA) y variables constantes detectadas en el EDA.
3. Corregir tipos de datos erróneos (fechas, códigos categóricos) y documentar cada conversión.
4. Generar variables derivadas que resuman información redundante (por ejemplo, número de diagnósticos registrados).
5. Exportar el resultado a un nuevo archivo Excel (`SaludMental_limpio.xlsx`) para su carga posterior en la base de datos.

0.4 Duplica el dataset original

```r
SaludMental_limpio <- SaludMental
```

Creamos una copia para trabajar de forma segura sin tocar el Excel de origen.

0.5 Columnas completamente nulas o constantes

```r
# Identificar columnas con 100% de NA
cols_todo_na <- SaludMental_limpio %>%
  summarise(across(everything(), ~ all(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "columna", values_to = "todo_na") %>%
  filter(todo_na) %>%
  pull(columna)

length(cols_todo_na)
```

```
[1] 26
```

```r
cols_todo_na
```

```
 [1] "CCAA Residencia"          "Procedimiento 12"
 [3] "Procedimiento 13"         "Procedimiento 14"
 [5] "Procedimiento 15"         "Procedimiento 16"
 [7] "Procedimiento 17"         "Procedimiento 18"
 [9] "Procedimiento 19"         "Procedimiento 20"
[11] "GDR AP"                   "CDM AP"
[13] "Tipo GDR AP"              "Valor Peso Español"
[15] "Tipo GDR APR"             "Valor Peso Americano APR"
[17] "Reingreso"                "GDR IR"
[19] "Tipo GDR IR"              "Tipo PROCESO IR"
[21] "Procedimiento Externo 1"  "Procedimiento Externo 2"
[23] "Procedimiento Externo 3"  "Procedimiento Externo 4"
[25] "Procedimiento Externo 5"  "Procedimiento Externo 6"
```

```r
# Identificar columnas constantes (un solo valor no nulo)
cols_constantes <- SaludMental_limpio %>%
  summarise(across(everything(), ~ n_distinct(na.omit(.)))) %>%
  pivot_longer(everything(), names_to = "columna", values_to = "n_dist") %>%
  filter(n_dist == 1) %>%
```

```
  pull(columna)

cols_constantes
```

```
[1] "CIE"
```

Estas listas confirman los hallazgos del EDA: columnas sin información útil. Procedemos a eliminarlas.

```
cols_a_eliminar <- union(cols_todo_na, cols_constantes)

cols_a_eliminar_existentes <- intersect(cols_a_eliminar, colnames(SaludMental_limpio))

SaludMental_limpio <- SaludMental_limpio %>%
  select(-all_of(cols_a_eliminar_existentes))

length(cols_a_eliminar_existentes)
```

```
[1] 27
```

```
ncol(SaludMental) - ncol(SaludMental_limpio)
```

```
[1] 27
```

Se eliminan todas las columnas redundantes presentes en el dataset, reduciendo el ancho de la tabla manteniendo solo atributos informativos.

0.6 Columnas con más del 95% de valores nulos

El EDA evidenció un número elevado de variables prácticamente vacías. Confirmamos qué columnas mantienen todavía más del 95% de `NA` y las descartamos para evitar ruido:

```
na_porcentaje <- SaludMental_limpio %>%
  summarise(across(everything(), ~ mean(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "columna", values_to = "pct_na") %>%
  mutate(pct_na = pct_na * 100) %>%
  arrange(desc(pct_na))
```

```r
cols_na95 <- na_porcentaje %>%
  filter(pct_na > 95) %>%
  pull(columna)

cols_na95
```

```
 [1] "Diagnóstico 20"        "POA Diagnóstico 20"    "Procedimiento 11"
 [4] "Diagnóstico 19"        "POA Diagnóstico 19"    "Procedimiento 10"
 [7] "Diagnóstico 18"        "POA Diagnóstico 18"    "Procedimiento 9"
[10] "Diagnóstico 17"        "POA Diagnóstico 17"    "Procedimiento 8"
[13] "Diagnóstico 16"        "POA Diagnóstico 16"    "Procedimiento 7"
[16] "Diagnóstico 15"        "POA Diagnóstico 15"    "Días UCI"
[19] "Procedimiento 6"       "Fecha de Intervención" "Diagnóstico 14"
[22] "POA Diagnóstico 14"    "Procedimiento 5"       "Diagnóstico 13"
[25] "POA Diagnóstico 13"    "Diagnóstico 12"        "POA Diagnóstico 12"
[28] "Procedimiento 4"       "Diagnóstico 11"        "POA Diagnóstico 11"
[31] "Diagnóstico 10"        "POA Diagnóstico 10"
```

```r
length(cols_na95)
```

```
[1] 32
```

```r
cols_na95_existentes <- intersect(cols_na95, colnames(SaludMental_limpio))

SaludMental_limpio <- SaludMental_limpio %>%
  select(-all_of(cols_na95_existentes))

length(cols_na95_existentes)
```

```
[1] 32
```

Eliminamos únicamente las columnas existentes con escasa información (más del 95% de valores nulos). Esto reduce la dimensionalidad y deja la tabla con atributos realmente útiles para análisis y modelado.

0.7 Normalización de columnas de fecha

Detectamos columnas temporales almacenadas como texto. Las convertimos a formatos de fecha apropiados:

```r
parse_fecha_flexible <- function(x) {
  if (inherits(x, "Date")) {
    return(x)
  }
  if (inherits(x, "POSIXt")) {
    return(as_date(x))
  }
  if (is.numeric(x)) {
    return(as_date(x, origin = "1899-12-30"))
  }
  x_chr <- trimws(as.character(x))
  x_chr[x_chr == ""] <- NA_character_
  parsed <- suppressWarnings(parse_date_time(
    x_chr,
    orders = c("Ymd", "dmY", "mdY", "Ymd HMS", "dmY HMS", "mdY HMS")
  ))
  as_date(parsed)
}

SaludMental_limpio <- SaludMental_limpio %>%
  mutate(
    `Fecha de Ingreso` = parse_fecha_flexible(`Fecha de Ingreso`),
    `Fecha de Fin Contacto` = parse_fecha_flexible(`Fecha de Fin Contacto`),
    `Mes de Ingreso` = suppressWarnings(as_date(paste0(`Mes de Ingreso`, "-01")))
  )

summary(select(SaludMental_limpio, `Fecha de Ingreso`, `Fecha de Fin Contacto`, `Mes de Ingre
```

```
 Fecha de Ingreso     Fecha de Fin Contacto Mes de Ingreso
 Min.   :2016-01-01   Min.   :2016-01-02    Min.   :2016-01-01
 1st Qu.:2016-10-02   1st Qu.:2016-10-17    1st Qu.:2016-10-01
 Median :2017-06-30   Median :2017-07-14    Median :2017-06-01
 Mean   :2017-07-04   Mean   :2017-07-19    Mean   :2017-06-19
 3rd Qu.:2018-04-13   3rd Qu.:2018-04-28    3rd Qu.:2018-04-01
 Max.   :2018-12-31   Max.   :2021-01-28    Max.   :2018-12-01
```

```r
fechas_no_parseadas <- SaludMental %>%
  transmute(
    fila = row_number(),
    `Fecha de Ingreso` = trimws(as.character(`Fecha de Ingreso`)),
    `Fecha de Fin Contacto` = trimws(as.character(`Fecha de Fin Contacto`))
```

```
  ) %>%
  filter(
    `Fecha de Ingreso` %in% c("", NA) | `Fecha de Fin Contacto` %in% c("", NA)
  )

nrow(fechas_no_parseadas)
```

```
[1] 0
```

```
head(fechas_no_parseadas, 5)
```

```
# A tibble: 0 x 3
# i 3 variables: fila <int>, Fecha de Ingreso <chr>,
#   Fecha de Fin Contacto <chr>
```

No hay valores mal formateados.

0.8 Normalización de `Fecha Inicio Contacto`

El extracto original almacena `Fecha Inicio Contacto` como una cadena compacta (por ejemplo, `01012016 1622`) que combina fecha y hora sin separadores. Para facilitar su análisis y cumplir con el estándar solicitado (`aaaa-mm-dd hh:mm`), separamos la información en dos columnas legibles: una para la fecha y otra para la hora.

```
normalizar_inicio_contacto <- function(x) {
  # Aseguramos que el vector esté en formato carácter y limpiamos entradas vacías
  x_chr <- trimws(as.character(x))
  x_chr[x_chr %in% c("", "NA")] <- NA_character_

  # Inicializamos vectores de salida
  fecha_formateada <- rep(NA_character_, length(x_chr))
  hora_formateada <- rep(NA_character_, length(x_chr))

  # Identificamos posiciones con información válida
  posiciones_validas <- which(!is.na(x_chr))

  if (length(posiciones_validas) > 0) {
    # Eliminamos cualquier separador y conservamos únicamente dígitos
    solo_digitos <- stringr::str_replace_all(x_chr[posiciones_validas], "\\D", "")
```

```
    # Extraemos la porción de fecha (primeros 8 dígitos) y hora (siguientes 4, si existen)
    fecha_bruta <- stringr::str_sub(solo_digitos, 1, 8)
    hora_bruta <- ifelse(stringr::str_length(solo_digitos) >= 12, stringr::str_sub(solo_digit

    # Parseamos la fecha usando el día como primera parte y la formateamos a dd/mm/aaaa
    fecha_parseada <- suppressWarnings(lubridate::dmy(fecha_bruta))
    fecha_formateada[posiciones_validas] <- ifelse(
      is.na(fecha_parseada),
      NA_character_,
      format(fecha_parseada, "%Y-%m-%d")
    )

    # Aseguramos el formato HH:MM para la hora
    hora_pad <- ifelse(is.na(hora_bruta), NA_character_, stringr::str_pad(hora_bruta, 4, pad
    hora_formateada[posiciones_validas] <- ifelse(
      is.na(hora_pad),
      NA_character_,
      paste0(stringr::str_sub(hora_pad, 1, 2), ":", stringr::str_sub(hora_pad, 3, 4))
    )
  }

  tibble(
    fecha_formateada = fecha_formateada,
    hora_formateada = hora_formateada
  )
}

columna_inicio_contacto <- intersect(
  c(
    "Fecha inicio contacto",
    "Fecha de inicio contacto",
    "Fecha de Inicio contacto",
    "Fecha Inicio Contacto"
  ),
  colnames(SaludMental_limpio)
)

if (length(columna_inicio_contacto) == 1) {
  valores_inicio <- normalizar_inicio_contacto(SaludMental_limpio[[columna_inicio_contacto]])

  SaludMental_limpio <- SaludMental_limpio %>%
    mutate(
```

```
      `Fecha Inicio Contacto Fecha` = valores_inicio$fecha_formateada,
      `Fecha Inicio Contacto Hora` = valores_inicio$hora_formateada
    ) %>%
    select(-all_of(columna_inicio_contacto)) %>%
    relocate(
      `Fecha Inicio Contacto Fecha`,
      `Fecha Inicio Contacto Hora`,
      .before = `Fecha de Fin Contacto`
    )
} else if (length(columna_inicio_contacto) > 1) {
  warning("Se han detectado múltiples columnas candidatas para 'Fecha Inicio Contacto'; revis
} else {
  warning("No se encontró la columna 'Fecha Inicio Contacto' en el dataset; verifique el nom
}
```

Este procedimiento preserva la trazabilidad temporal del contacto clínico, evita ambigüedades en el parseo y habilita tanto filtros por fecha como análisis intradía sin depender de transformaciones adicionales.

0.9 Limpieza de texto en columnas categóricas relevantes

Alineamos mayúsculas/minúsculas y espacios para evitar duplicados artificiales:

```
columnas_texto <- c(
  "Comunidad Autónoma",
  "Diagnóstico Principal",
  "Procedimiento 1",
  "Centro Recodificado",
  "País Nacimiento"
)

SaludMental_limpio <- SaludMental_limpio %>%
  mutate(across(all_of(columnas_texto), ~ str_squish(str_to_upper(as.character(.)))))
```

Esto garantiza que valores como “andalucía”, “Andalucía” y “ ANDALUCIA ” se unifiquen.

0.10 Columnas de diagnósticos y procedimientos

```r
cols_diagnosticos <- names(SaludMental_limpio)[str_detect(names(SaludMental_limpio), "^Diagn
cols_procedimientos <- names(SaludMental_limpio)[str_detect(names(SaludMental_limpio), "^Proc

cols_diagnosticos
```

```
[1] "Diagnóstico Principal" "Diagnóstico 2"         "Diagnóstico 3"
[4] "Diagnóstico 4"         "Diagnóstico 5"         "Diagnóstico 6"
[7] "Diagnóstico 7"         "Diagnóstico 8"         "Diagnóstico 9"
```

```r
cols_procedimientos
```

```
[1] "Procedimiento 1" "Procedimiento 2" "Procedimiento 3"
```

Estas listas incluyen el diagnóstico principal y los secundarios, además de los procedimientos registrados por fila.

0.11 Variables derivadas

```r
SaludMental_limpio <- SaludMental_limpio %>%
  mutate(
    Diagnosticos_totales = rowSums(!is.na(select(., all_of(cols_diagnosticos))), na.rm = TRU
    Procedimientos_totales = rowSums(!is.na(select(., all_of(cols_procedimientos))), na.rm =
    Tiene_procedimiento = Procedimientos_totales > 0,
    Diagnostico_F = str_starts(`Diagnóstico Principal`, "F"),
    Diagnostico_F = replace_na(Diagnostico_F, FALSE),
    Tiene_Comorbilidad = Diagnosticos_totales > 1,
    Duracion_Episodio_Calculada = as.integer(`Fecha de Fin Contacto` - `Fecha de Ingreso`)
  )

summary(select(SaludMental_limpio, Diagnosticos_totales, Procedimientos_totales, Tiene_proce
```

```
 Diagnosticos_totales Procedimientos_totales Tiene_procedimiento Diagnostico_F
 Min.   :1.00         Min.   :0.0000         Mode :logical       Mode:logical
 1st Qu.:2.00         1st Qu.:0.0000         FALSE:16590         TRUE:21210
 Median :4.00         Median :0.0000         TRUE :4620
 Mean   :4.17         Mean   :0.3971
 3rd Qu.:6.00         3rd Qu.:0.0000
 Max.   :9.00         Max.   :3.0000
```

```
 Tiene_Comorbilidad Duracion_Episodio_Calculada
 Mode :logical      Min.   :  0.00
 FALSE:2604         1st Qu.:  5.00
 TRUE :18606        Median : 11.00
                    Mean   : 15.34
                    3rd Qu.: 19.00
                    Max.   :814.00
```

`Diagnosticos_totales` y `Procedimientos_totales` cuantifican la carga clínica; `Tiene_procedimiento` ayuda a segmentar filas sin intervención registrada; `Diagnostico_F` identifica patologías mentales (códigos F en CIE-10); `Tiene_Comorbilidad` marca episodios con múltiples diagnósticos y `Duracion_Episodio_Calculada` estima la duración del contacto en días.

0.12 Eliminación de columnas redundantes

Verificamos que `Edad` y `Edad en Ingreso` son equivalentes y descartamos la duplicada:

```
identicas <- identical(SaludMental_limpio$Edad, SaludMental_limpio$`Edad en Ingreso`)
identicas
```

```
[1] FALSE
```

```
if (identicas) {
  SaludMental_limpio <- SaludMental_limpio %>%
    select(-Edad)
}
```

De esta forma conservamos únicamente `Edad en Ingreso`, que aporta la misma información.

0.13 Segmentación por grupo etario

Construimos una variable categórica basada en `Edad en Ingreso` para análisis demográficos:

```
SaludMental_limpio <- SaludMental_limpio %>%
  mutate(
    Grupo_Etario = case_when(
      `Edad en Ingreso` < 18 ~ "0-17",
      `Edad en Ingreso` >= 18 & `Edad en Ingreso` <= 35 ~ "18-35",
      `Edad en Ingreso` >= 36 & `Edad en Ingreso` <= 64 ~ "36-64",
```

```
      `Edad en Ingreso` >= 65 ~ "65+",
      TRUE ~ NA_character_
    )
  )

table(SaludMental_limpio$Grupo_Etario, useNA = "ifany")
```

```
 0-17 18-35 36-64   65+
  538  5516 13755  1401
```

`Grupo_Etario` agrupa a los pacientes en bloques interpretables (menores, jóvenes, adultos y seniors) sin alterar la edad continua.

0.14 Componentes temporales del ingreso

Derivamos el nombre del día y del mes de `Fecha de Ingreso` para análisis estacionales:

```
dias_semana <- c("lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo")
meses_nombre <- c("enero", "febrero", "marzo", "abril", "mayo", "junio", "julio", "agosto",

SaludMental_limpio <- SaludMental_limpio %>%
  mutate(
    Dia_Semana_Ingreso = factor(dias_semana[wday(`Fecha de Ingreso`, week_start = 1)], levels
    Mes_Nombre_Ingreso = factor(meses_nombre[month(`Fecha de Ingreso`)], levels = meses_nombr
  )

table(SaludMental_limpio$Dia_Semana_Ingreso, useNA = "ifany")
```

```
   lunes    martes miercoles    jueves   viernes    sabado   domingo
    3483      3556      3370      3424      3284      2083      2010
```

Estas variables permiten estudiar patrones semanales y mensuales sin modificar la fecha original.

0.15 Tratamiento de valores atípicos en `Estancia Días`

Aplicamos winsorización en el percentil 99 para acotar estancias extremas sin perder la referencia original:

```
percentil_99_estancia <- quantile(SaludMental_limpio$`Estancia Días`, 0.99, na.rm = TRUE)
percentil_99_estancia
```

```
99%
 81
```

```
SaludMental_limpio <- SaludMental_limpio %>%
  mutate(
    Estancia_Dias_Acotada = pmin(`Estancia Días`, percentil_99_estancia)
  )
```

La columna `Estancia_Dias_Acotada` facilita análisis robustos mientras se preserva la serie original.

0.16 Resumen de hallazgos del EDA

En el análisis exploratorio identificamos varios focos de limpieza:

- Más de 25 columnas con 100% de valores nulos (`Procedimiento 12`-`Procedimiento 20`, `GDR IR`, `CCAA Residencia`, etc.).
- Columnas numéricas que en realidad codifican categorías (`Sexo`, `Tipo Alta`, `Nivel Severidad APR`, `Riesgo Mortalidad APR`, `GRD APR`, `CDM APR`).
- Campos temporales almacenados como texto (`Fecha de Fin Contacto`, `Mes de Ingreso`).
- Variables redundantes (`Edad` y `Edad en Ingreso`) y constantes (`CIE` siempre igual a 10).
- Alta tasa de nulos estructurales en diagnósticos/procedimientos secundarios, susceptibles de agregarse en contadores.

Utilizamos estos hallazgos para diseñar la estrategia de feature engineering sin modificar el archivo original.

0.17 Validación del dataset limpio

```r
tibble(
  filas_original = nrow(SaludMental),
  columnas_original = ncol(SaludMental),
  filas_limpio = nrow(SaludMental_limpio),
  columnas_limpio = ncol(SaludMental_limpio)
)
```

```
# A tibble: 1 x 4
  filas_original columnas_original filas_limpio columnas_limpio
           <int>             <int>        <int>           <int>
1          21210               111        21210              63
```

```r
top_na_limpio <- SaludMental_limpio %>%
  summarise(across(everything(), ~ sum(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "columna", values_to = "na") %>%
  arrange(desc(na)) %>%
  head(10)

top_na_limpio
```

```
# A tibble: 10 x 2
   columna              na
   <chr>             <int>
 1 Procedimiento 3   20136
 2 Diagnóstico 9     19593
 3 POA Diagnóstico 9 19593
 4 Diagnóstico 8     18686
 5 POA Diagnóstico 8 18686
 6 Procedimiento 2   18482
 7 Diagnóstico 7     17375
 8 POA Diagnóstico 7 17375
 9 Procedimiento 1   16590
10 Diagnóstico 6     15447
```

El recuento confirma que no se eliminaron filas y que la reducción de columnas es la esperada. Los nulos restantes provienen de ausencias estructurales o conversiones fallidas en fechas (revisar `Fecha de Fin Contacto` y `Mes de Ingreso`).

0.18 Anonimización

En cumplimiento de los requisitos de gobernanza de datos sanitarios y para proteger la identidad de las personas atendidas, aplicamos un proceso de anonimización reversible. Consiste en separar la información identificativa en una tabla aislada y sustituir granjas de pacientes por identificadores opacos.

```
# Generamos una tabla de correspondencia con UUID y datos identificativos mínimos
pacientes_anonimos <- SaludMental %>%
  select(`Número de registro anual`, `Nombre`, `Fecha de nacimiento`) %>%
  distinct(`Número de registro anual`, .keep_all = TRUE) %>%
  mutate(
    UUID_Paciente = replicate(n(), UUIDgenerate())
  ) %>%
  select(UUID_Paciente, `Número de registro anual`, `Nombre`, `Fecha de nacimiento`)

# Vinculamos las UUID con el dataset limpio para crear un dataframe anonimizado
SaludMental_limpio_anon <- SaludMental_limpio %>%
  left_join(pacientes_anonimos %>% select(UUID_Paciente, `Número de registro anual`), by = "
  mutate(UUID_Paciente = replace_na(UUID_Paciente, UUIDgenerate())) %>%
  select(-`Nombre`) %>%
  relocate(UUID_Paciente, .after = `Número de registro anual`)

# Guardamos la tabla de correspondencia para custodia segura
write_xlsx(pacientes_anonimos, "../pacientes_anonimos.xlsx")
#write_xlsx(SaludMental_limpio_anon, "../SaludMental_limpio_anon.xlsx")
```

La tabla `pacientes_anonimos` concentra únicamente los datos identificativos esenciales (`Nombre` y `Fecha de Nacimiento`) ligados a una `UUID_Paciente` generada con la librería `uuid`. Esta correspondencia se mantiene en un fichero separado (`pacientes_anonimos.xlsx`) que debe resguardarse bajo controles de acceso estrictos. Por su parte, `SaludMental_limpio_anon` sustituye el nombre del paciente por la UUID en la tabla operativa (`SaludMental_limpio_anon.xlsx`).

De este modo, la base de datos final que se distribuya para análisis permanece anonimizada, evitando exposiciones directas de datos personales, mientras que la reversibilidad se preserva mediante la tabla de correspondencia custodiada y auditada. Esto cumple las exigencias del reto y posibilita restaurar la identidad cuando exista base legal y autorización expresa.

El archivo `SaludMental_limpio.xlsx` queda listo para carga en la base de datos; el Excel original permanece intacto.