# Manipulacion de datos con R

Anderson Ocaña

2025-08-05

## Contents



### 0.1 Intalacion de paquetes si es necesario

```
if (!require("tidyverse")) install.packages("tidyverse")
```

```
## Loading required package: 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   3.5.2     v tibble    3.3.0
## 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 become errors
```

#### Carga de libreras

```
library(tidyverse)
```

### Obtencion de datos por medio de una fuente URL

```
url_path <- "https://gist.githubusercontent.com/rnirmal/e01acfdaf54a6f9b24e91ba4cae63518/raw/6b589a5c5a8
data <- read.csv(url_path, sep = ",")

glimpse(data)
```

```
## Rows: 61
## Columns: 6
## $ datasetName  <chr> "Microbiome Project", "GloBI", "Global Climate", "CommonC~
## $ about        <chr> "American Gut (Microbiome Project)", "Global Biotic Inter~
## $ link         <chr> "https://github.com/biocore/American-Gut", "https://githu~
## $ categoryName <chr> "Biology", "Biology", "Climate/Weather", "Computer Networ~
## $ cloud        <chr> "GitHub", "GitHub", "", "", "", "", "", "", "GitHub", "",~
## $ vintage      <int> NA, NA, 1929, 2012, NA, NA, 2009, 2012, NA, NA, NA, NA, N~
```

### Obtencion de datos por medio de un archivo local

```
library(dplyr)
library(here)
```

```
## here() starts at /home/anderson/Documents/data-science-r-section
```

```
string_path <- "./R-Clases/week-1/clase_3/csv/data_sample.csv"

csv_path <- string_path %>%
  strsplit(split = "/", fixed = TRUE) %>%
  .[[1]] %>%
  {
    do.call(here::here, as.list(.))
  }

if (file.exists(csv_path)) {
  groups <- read.csv(csv_path)
  head(groups)
} else {
  warning("No existe el archivo")
}
```

```
##         id                                                name
## 1 22837760                           WooCommerce Madrid Meetup
## 2 18313218                              Cloud Computing Meetup
## 3 10979332 Gr2Dest - Grupo de estudio de seguridad informática
## 4 19641353          Rebeldes del Sector TI por la Conciliación
## 5 19136010                                   TensorFlow Madrid
## 6 20030165                              Bot Development Madrid
##                                 urlname
## 1                 WooCommerce-Madrid-Meetup
## 2                     Cloud-Computing-Spain
## 3                                 Gr2Dest
## 4 Rebeldes-del-Sector-TI-por-la-Conciliacion
## 5                         TensorFlow-Madrid
## 6                    Bot-Development-Madrid
##                                                          link rating
## 1                 https://www.meetup.com/WooCommerce-Madrid-Meetup/   0.00
## 2                     https://www.meetup.com/Cloud-Computing-Spain/   4.51
## 3                                 https://www.meetup.com/Gr2Dest/   4.94
```

```
## 4 https://www.meetup.com/Rebeldes-del-Sector-TI-por-la-Conciliacion/   4.80
## 5                          https://www.meetup.com/TensorFlow-Madrid/   5.00
## 6                     https://www.meetup.com/Bot-Development-Madrid/   4.57
##        created
## 1 1.489164e+12
## 2 1.420730e+12
## 3 1.383661e+12
## 4 1.456744e+12
## 5 1.447918e+12
## 6 1.464974e+12
##
## 1 <p>## Español ##</p>\n<p>WooCommerce se ha convertido en el método más popular para vender online,
## 2
## 3
## 4
## 5                                              <p><img src="http://photos3.meetupstatic.com/photos/even
## 6
##          organiserName organiserMemberId
## 1          WooCommerce         168398372
## 2 José Ignacio Herranz          69788672
## 3             R3D3N3MY          70320852
## 4     Susana Morcuende         196953445
## 5                Renat         195761504
## 6        Rafael Casuso          28421072
```

**Obtencion de datos por medio de una base de datos**

```
# install dependencies
install.packages("DBI")
```

```
## Installing package into '/home/anderson/R/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
install.packages("RMySQL")
```

```
## Installing package into '/home/anderson/R/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
# library(DBI)
library(RMySQL)
```

```
## Loading required package: DBI
```

**Obtencion de datos por medio de una consulta**

```
# Construcción de la consulta SQL
query_data <- paste(
  "SELECT",
  "    c.id,",
  "    c.ozone,",
  "    c.solar_r,",
  "    c.wind,",
  "    c.temp,",
  "    c.month,",
  "    c.day,",
  "    w.name",
  "FROM `weather`.`cities` AS w",
  "LEFT JOIN `weather`.`weather_data` AS c ON w.id = c.city_id",
```

```
  sep = "\n"
)

# Conexión a la base de datos MySQL
tryCatch(
  {
    mysql_cnx <- dbConnect(
      MySQL(),
      dbname   = "weather",
      host     = "127.0.0.1",
      port     = 3306,
      user     = "root",
      password = "pass_admin"
    )

    # Ejecutar consulta y obtener resultados como dataframe
    dataframe <- dbGetQuery(mysql_cnx, query_data)

    dbDisconnect(mysql_cnx)

    # Ver las primeras filas del dataframe
    head(dataframe)
  },
  error = function(e) {
    message("No es posible conectarse a la base de datos")
    message(e$message)
  }
)
```

```
## No es posible conectarse a la base de datos
```

```
## Failed to connect to database: Error: Can't connect to server on '127.0.0.1' (115)
```

---

## 0.2 Limpieza de datos

### Un poco de data sobre el dataset a usar

```
data(airquality)
summary(airquality)
```

```
##      Ozone           Solar.R          Wind             Temp
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
##  Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
##  NA's   :37       NA's   :7
##      Month            Day
##  Min.   :5.000   Min.   : 1.0
##  1st Qu.:6.000   1st Qu.: 8.0
##  Median :7.000   Median :16.0
##  Mean   :6.993   Mean   :15.8
##  3rd Qu.:8.000   3rd Qu.:23.0
```

```
##  Max.   :9.000   Max.   :31.0
##
```

```
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
names(airquality)
```

```
## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

- **Formas de selecion de la data dentro del dataset**

```
print("Realizando seleccionado por codigo vectorial")
```

```
## [1] "Realizando seleccionado por codigo vectorial"
```

```
seleccion <- airquality$Month == 5

mayo <- airquality[seleccion, ]
junio <- airquality[airquality$Month == 6, ]

mayo
```

```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     NA      NA 14.3   56     5   5
## 6     28      NA 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    NA     194  8.6   69     5  10
## 11     7      NA  6.9   74     5  11
## 12    16     256  9.7   69     5  12
## 13    11     290  9.2   66     5  13
## 14    14     274 10.9   68     5  14
## 15    18      65 13.2   58     5  15
## 16    14     334 11.5   64     5  16
## 17    34     307 12.0   66     5  17
## 18     6      78 18.4   57     5  18
## 19    30     322 11.5   68     5  19
## 20    11      44  9.7   62     5  20
## 21     1       8  9.7   59     5  21
## 22    11     320 16.6   73     5  22
## 23     4      25  9.7   61     5  23
## 24    32      92 12.0   61     5  24
## 25    NA      66 16.6   57     5  25
## 26    NA     266 14.9   58     5  26
```

```
## 27    NA      NA  8.0   57     5  27
## 28    23      13 12.0   67     5  28
## 29    45     252 14.9   81     5  29
## 30   115     223  5.7   79     5  30
## 31    37     279  7.4   76     5  31
```

```
junio
```

```
##    Ozone Solar.R Wind Temp Month Day
## 32    NA     286  8.6   78     6   1
## 33    NA     287  9.7   74     6   2
## 34    NA     242 16.1   67     6   3
## 35    NA     186  9.2   84     6   4
## 36    NA     220  8.6   85     6   5
## 37    NA     264 14.3   79     6   6
## 38    29     127  9.7   82     6   7
## 39    NA     273  6.9   87     6   8
## 40    71     291 13.8   90     6   9
## 41    39     323 11.5   87     6  10
## 42    NA     259 10.9   93     6  11
## 43    NA     250  9.2   92     6  12
## 44    23     148  8.0   82     6  13
## 45    NA     332 13.8   80     6  14
## 46    NA     322 11.5   79     6  15
## 47    21     191 14.9   77     6  16
## 48    37     284 20.7   72     6  17
## 49    20      37  9.2   65     6  18
## 50    12     120 11.5   73     6  19
## 51    13     137 10.3   76     6  20
## 52    NA     150  6.3   77     6  21
## 53    NA      59  1.7   76     6  22
## 54    NA      91  4.6   76     6  23
## 55    NA     250  6.3   76     6  24
## 56    NA     135  8.0   75     6  25
## 57    NA     127  8.0   78     6  26
## 58    NA      47 10.3   73     6  27
## 59    NA      98 11.5   80     6  28
## 60    NA      31 14.9   77     6  29
## 61    NA     138  8.0   83     6  30
```

# 1 Función subset()

subset(x=, subset=, select=)

Los parámetros de esta función son:

- `x` indica el nombre del data frame al que queremos aplicar el filtro
- `subset` es una expresión lógica que indica las filas que queremos conservar
- `select` es un vector indicando los nombres de las columnas que queremos conservar

```
julio <- subset(
  x = airquality,
  select = -c(Month, Day, Temp), # seleccionamos un grupo de columnas
  subset = Month == 7
) # seleccionamos las filas donde el mes es julio
julio
```

```
##    Ozone Solar.R Wind
## 62   135     269  4.1
## 63    49     248  9.2
## 64    32     236  9.2
## 65    NA     101 10.9
## 66    64     175  4.6
## 67    40     314 10.9
## 68    77     276  5.1
## 69    97     267  6.3
## 70    97     272  5.7
## 71    85     175  7.4
## 72    NA     139  8.6
## 73    10     264 14.3
## 74    27     175 14.9
## 75    NA     291 14.9
## 76     7      48 14.3
## 77    48     260  6.9
## 78    35     274 10.3
## 79    61     285  6.3
## 80    79     187  5.1
## 81    63     220 11.5
## 82    16       7  6.9
## 83    NA     258  9.7
## 84    NA     295 11.5
## 85    80     294  8.6
## 86   108     223  8.0
## 87    20      81  8.6
## 88    52      82 12.0
## 89    82     213  7.4
## 90    50     275  7.4
## 91    64     253  7.4
## 92    59     254  9.2
```

## 2 Selecciones aleatorias

sample(x=, size=, replace=)

- `x` vector de donde obtener las muestras
- `size` tamaño de la muestra seleccionada
- `replace` es un vector lógico que indica si la muestra es con reposición. Si es *FALSE* los elementos pueden ser seleccionados una sola vez. Si es *TRUE* los elementos pueden ser seleccionados repetidas veces.

```
names(airquality)
```

```
## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
```

```
sample(x = airquality %>% filter("Solar.R" > 100), size = 5, replace = TRUE)
```

```
##     Wind Temp Month Wind.1 Day
## 1    7.4   67     5    7.4   1
## 2    8.0   72     5    8.0   2
## 3   12.6   74     5   12.6   3
## 4   11.5   62     5   11.5   4
```

```
## 5   14.3   56     5   14.3   5
## 6   14.9   66     5   14.9   6
## 7    8.6   65     5    8.6   7
## 8   13.8   59     5   13.8   8
## 9   20.1   61     5   20.1   9
## 10   8.6   69     5    8.6  10
## 11   6.9   74     5    6.9  11
## 12   9.7   69     5    9.7  12
## 13   9.2   66     5    9.2  13
## 14  10.9   68     5   10.9  14
## 15  13.2   58     5   13.2  15
## 16  11.5   64     5   11.5  16
## 17  12.0   66     5   12.0  17
## 18  18.4   57     5   18.4  18
## 19  11.5   68     5   11.5  19
## 20   9.7   62     5    9.7  20
## 21   9.7   59     5    9.7  21
## 22  16.6   73     5   16.6  22
## 23   9.7   61     5    9.7  23
## 24  12.0   61     5   12.0  24
## 25  16.6   57     5   16.6  25
## 26  14.9   58     5   14.9  26
## 27   8.0   57     5    8.0  27
## 28  12.0   67     5   12.0  28
## 29  14.9   81     5   14.9  29
## 30   5.7   79     5    5.7  30
## 31   7.4   76     5    7.4  31
## 32   8.6   78     6    8.6   1
## 33   9.7   74     6    9.7   2
## 34  16.1   67     6   16.1   3
## 35   9.2   84     6    9.2   4
## 36   8.6   85     6    8.6   5
## 37  14.3   79     6   14.3   6
## 38   9.7   82     6    9.7   7
## 39   6.9   87     6    6.9   8
## 40  13.8   90     6   13.8   9
## 41  11.5   87     6   11.5  10
## 42  10.9   93     6   10.9  11
## 43   9.2   92     6    9.2  12
## 44   8.0   82     6    8.0  13
## 45  13.8   80     6   13.8  14
## 46  11.5   79     6   11.5  15
## 47  14.9   77     6   14.9  16
## 48  20.7   72     6   20.7  17
## 49   9.2   65     6    9.2  18
## 50  11.5   73     6   11.5  19
## 51  10.3   76     6   10.3  20
## 52   6.3   77     6    6.3  21
## 53   1.7   76     6    1.7  22
## 54   4.6   76     6    4.6  23
## 55   6.3   76     6    6.3  24
## 56   8.0   75     6    8.0  25
## 57   8.0   78     6    8.0  26
## 58  10.3   73     6   10.3  27
```

```
## 59  11.5   80     6   11.5  28
## 60  14.9   77     6   14.9  29
## 61   8.0   83     6    8.0  30
## 62   4.1   84     7    4.1   1
## 63   9.2   85     7    9.2   2
## 64   9.2   81     7    9.2   3
## 65  10.9   84     7   10.9   4
## 66   4.6   83     7    4.6   5
## 67  10.9   83     7   10.9   6
## 68   5.1   88     7    5.1   7
## 69   6.3   92     7    6.3   8
## 70   5.7   92     7    5.7   9
## 71   7.4   89     7    7.4  10
## 72   8.6   82     7    8.6  11
## 73  14.3   73     7   14.3  12
## 74  14.9   81     7   14.9  13
## 75  14.9   91     7   14.9  14
## 76  14.3   80     7   14.3  15
## 77   6.9   81     7    6.9  16
## 78  10.3   82     7   10.3  17
## 79   6.3   84     7    6.3  18
## 80   5.1   87     7    5.1  19
## 81  11.5   85     7   11.5  20
## 82   6.9   74     7    6.9  21
## 83   9.7   81     7    9.7  22
## 84  11.5   82     7   11.5  23
## 85   8.6   86     7    8.6  24
## 86   8.0   85     7    8.0  25
## 87   8.6   82     7    8.6  26
## 88  12.0   86     7   12.0  27
## 89   7.4   88     7    7.4  28
## 90   7.4   86     7    7.4  29
## 91   7.4   83     7    7.4  30
## 92   9.2   81     7    9.2  31
## 93   6.9   81     8    6.9   1
## 94  13.8   81     8   13.8   2
## 95   7.4   82     8    7.4   3
## 96   6.9   86     8    6.9   4
## 97   7.4   85     8    7.4   5
## 98   4.6   87     8    4.6   6
## 99   4.0   89     8    4.0   7
## 100 10.3   90     8   10.3   8
## 101  8.0   90     8    8.0   9
## 102  8.6   92     8    8.6  10
## 103 11.5   86     8   11.5  11
## 104 11.5   86     8   11.5  12
## 105 11.5   82     8   11.5  13
## 106  9.7   80     8    9.7  14
## 107 11.5   79     8   11.5  15
## 108 10.3   77     8   10.3  16
## 109  6.3   79     8    6.3  17
## 110  7.4   76     8    7.4  18
## 111 10.9   78     8   10.9  19
## 112 10.3   78     8   10.3  20
```

```
## 113 15.5   77     8   15.5  21
## 114 14.3   72     8   14.3  22
## 115 12.6   75     8   12.6  23
## 116  9.7   79     8    9.7  24
## 117  3.4   81     8    3.4  25
## 118  8.0   86     8    8.0  26
## 119  5.7   88     8    5.7  27
## 120  9.7   97     8    9.7  28
## 121  2.3   94     8    2.3  29
## 122  6.3   96     8    6.3  30
## 123  6.3   94     8    6.3  31
## 124  6.9   91     9    6.9   1
## 125  5.1   92     9    5.1   2
## 126  2.8   93     9    2.8   3
## 127  4.6   93     9    4.6   4
## 128  7.4   87     9    7.4   5
## 129 15.5   84     9   15.5   6
## 130 10.9   80     9   10.9   7
## 131 10.3   78     9   10.3   8
## 132 10.9   75     9   10.9   9
## 133  9.7   73     9    9.7  10
## 134 14.9   81     9   14.9  11
## 135 15.5   76     9   15.5  12
## 136  6.3   77     9    6.3  13
## 137 10.9   71     9   10.9  14
## 138 11.5   71     9   11.5  15
## 139  6.9   78     9    6.9  16
## 140 13.8   67     9   13.8  17
## 141 10.3   76     9   10.3  18
## 142 10.3   68     9   10.3  19
## 143  8.0   82     9    8.0  20
## 144 12.6   64     9   12.6  21
## 145  9.2   71     9    9.2  22
## 146 10.3   81     9   10.3  23
## 147 10.3   69     9   10.3  24
## 148 16.6   63     9   16.6  25
## 149  6.9   70     9    6.9  26
## 150 13.2   77     9   13.2  27
## 151 14.3   75     9   14.3  28
## 152  8.0   76     9    8.0  29
## 153 11.5   68     9   11.5  30
```

---

# 3 Manipulación y Limpieza de Datos en R

## 3.1 Introducción

La limpieza de datos es un paso fundamental en el análisis de datos que puede representar hasta el 80% del tiempo total de un proyecto. Un dataset limpio es la base para obtener resultados confiables y análisis precisos.

```r
source(here::here("./R-Clases/week-1/clase_3/utilidades/creacion_dataset.R"))

df <- generar_dataset_ejemplo(n = 150, show_summary = TRUE)
```

```
## Dataset generado con 158 filas y 13 columnas
## Dimensiones: 158 13
##
## === RESUMEN DE PROBLEMAS EN EL DATASET ===
## Total de NA por columna:
##            id        nombre          edad salario_texto  departamento
##             3            18            18             5            11
## fecha_ingreso         email      telefono        genero  estado_civil
##            15            19             9            22            25
##   experiencia       puntaje         notas
##            15            20            13
##
## Ejemplos de problemas específicos:
## - Nombres con formato inconsistente: 77 casos
## - Salarios en formato texto: 153 casos
## - Edades fuera de rango (0-100): 3 casos
## - IDs duplicados: 15 casos
## - Filas completamente duplicadas: 7 casos
```

```
library(dplyr)
glimpse(df)
```

```
## Rows: 158
## Columns: 13
## $ id            <int> 136, 116, 58, 10, 13, 117, 31, 43, 119, 127, 5, 24, 86, ~
## $ nombre        <chr> NA, "  Isabel Fernández  ", "Carmen  Rodríguez", NA, "AN~
## $ edad          <dbl> NA, 19, 58, NA, 62, 30, 42, 46, 20, 150, 61, 31, 63, 35,~
## $ salario_texto <chr> "$ 35078", "$68209", "$78433", "$ 67101", "$72220", "$36~
## $ departamento  <chr> "RRHH", "Sistemas", "MARKETING", "finanzas", "Ventas", "~
## $ fecha_ingreso <chr> NA, "16/05/2020", "24/04/2021", "", "16/07/2020", "19/01~
## $ email         <chr> NA, "maria628@empresa.com", "maria967@gmail.com", NA, "m~
## $ telefono      <chr> "+541130448368", "011-7381-6062", "011-7214-3197", "011-~
## $ genero        <chr> "M", NA, "M", NA, "Femenino", "Mujer", NA, "f", "H", "M"~
## $ estado_civil  <chr> "Soltero", "Soltero", "Divorciado", "Divorciado", "Solte~
## $ experiencia   <dbl> -2, 4, 6, NA, 1, 1, 28, 5, 25, 14, 15, 31, 38, 9, 5, 11,~
## $ puntaje       <dbl> NA, 6, 5, NA, 1, 4, 7, 9, 5, 5, 3, 2, 8, 9, 10, 8, 4, 4,~
## $ notas         <chr> "promedio", "Sin comentarios", "NECESITA MEJORAS", NA, "~
```

## 3.2 1. Inspección Inicial de Datos

### 3.2.1 1.1 Exploración Básica

Antes de limpiar los datos, es esencial entender su estructura y contenido.

```
# Inspección básica
str(df)           # Estructura del dataframe
```

```
## 'data.frame':    158 obs. of  13 variables:
##  $ id           : int  136 116 58 10 13 117 31 43 119 127 ...
##  $ nombre       : chr  NA "  Isabel Fernández  " "Carmen  Rodríguez" NA ...
##  $ edad         : num  NA 19 58 NA 62 30 42 46 20 150 ...
##  $ salario_texto: chr  "$ 35078" "$68209" "$78433" "$ 67101" ...
##  $ departamento : chr  "RRHH" "Sistemas" "MARKETING" "finanzas" ...
##  $ fecha_ingreso: chr  NA "16/05/2020" "24/04/2021" "" ...
##  $ email        : chr  NA "maria628@empresa.com" "maria967@gmail.com" NA ...
```

```
##  $ telefono     : chr  "+541130448368" "011-7381-6062" "011-7214-3197" "011-1234" ...
##  $ genero       : chr  "M" NA "M" NA ...
##  $ estado_civil : chr  "Soltero" "Soltero" "Divorciado" "Divorciado" ...
##  $ experiencia  : num  -2 4 6 NA 1 1 28 5 25 14 ...
##  $ puntaje      : num  NA 6 5 NA 1 4 7 9 5 5 ...
##  $ notas        : chr  "promedio" "Sin comentarios" "NECESITA MEJORAS" NA ...
```

```
head(df)          # Primeras 6 filas
```

```
##    id               nombre edad salario_texto departamento fecha_ingreso
## 1 136                 <NA>   NA       $ 35078         RRHH          <NA>
## 2 116   Isabel Fernández     19        $68209     Sistemas    16/05/2020
## 3  58    Carmen  Rodríguez   58        $78433    MARKETING    24/04/2021
## 4  10                 <NA>   NA       $ 67101     finanzas
## 5  13            ANA MARÍA   62        $72220       Ventas    16/07/2020
## 6 117         maría garcía   30        $36311     finanzas    19/01/2023
##                  email      telefono   genero estado_civil experiencia puntaje
## 1                 <NA> +541130448368        M      Soltero          -2      NA
## 2 maria628@empresa.com 011-7381-6062     <NA>      Soltero           4       6
## 3   maria967@gmail.com 011-7214-3197        M   Divorciado           6       5
## 4                 <NA>      011-1234     <NA>   Divorciado          NA      NA
## 5   maria174@gmail.com 011-1268-2898 Femenino      Soltero           1       1
## 6 maria367@empresa.com 011-3986-5862    Mujer         <NA>           1       4
##               notas
## 1           promedio
## 2    Sin comentarios
## 3   NECESITA MEJORAS
## 4               <NA>
## 5   Buen desempeño
## 6
```

```
tail(df)          # Últimas 6 filas
```

```
##     id          nombre edad salario_texto     departamento fecha_ingreso
## 153  1   SOFIA TORRES    57        $49418 Recursos Humanos    14/08/2018
## 154 81    maría garcía   34        $74989 Recursos Humanos    17/05/2018
## 155 26   SOFIA TORRES    40        $52924             rrhh    02/05/2022
## 156 64       ANA MARÍA   44        $68466         Sistemas    25/07/2015
## 157 23       ANA MARÍA   62        $76437        Marketing    31/03/2016
## 158 50    José Antonio   24        $55930             rrhh    14/07/2015
##                    email      telefono    genero estado_civil experiencia
## 153  maria754@empresa.com 011-8132-8730 Masculino   Divorciado          38
## 154  pedro916@empresa.com 011-4742-2177  Femenino      Soltero          34
## 155   carlos217@gmail.com 011-3095-1074         m       Casado          22
## 156 carlos728@hotmail.com 011-8394-2822         m   Divorciado          16
## 157       ana56@gmail.com 011-2766-5905         H   Divorciado          12
## 158  pedro601@hotmail.com 011-6458-4435         f      Soltero           7
##     puntaje             notas
## 153       1   Buen desempeño
## 154       2           promedio
## 155       3           promedio
## 156       1    Sin comentarios
## 157       9   NECESITA MEJORAS
## 158       9    Sin comentarios
```

```r
summary(df)       # Resumen estadístico
```

```
##        id            nombre               edad        salario_texto
##  Min.   :  1.00   Length:158         Min.   : -5.00   Length:158
##  1st Qu.: 24.50   Class :character   1st Qu.: 26.75   Class :character
##  Median : 63.00   Mode  :character   Median : 40.50   Mode  :character
##  Mean   : 64.15                      Mean   : 46.56
##  3rd Qu.:101.50                      3rd Qu.: 52.25
##  Max.   :140.00                      Max.   :999.00
##  NA's   :3                           NA's   :18
##  departamento       fecha_ingreso         email             telefono
##  Length:158         Length:158          Length:158         Length:158
##  Class :character   Class :character    Class :character   Class :character
##  Mode  :character   Mode  :character    Mode  :character   Mode  :character
##
##
##
##
##     genero          estado_civil        experiencia        puntaje
##  Length:158         Length:158          Min.   : -2.00   Min.   :-1.000
##  Class :character   Class :character    1st Qu.:  9.00   1st Qu.: 3.000
##  Mode  :character   Mode  :character    Median : 19.00   Median : 5.000
##                                         Mean   : 20.36   Mean   : 5.471
##                                         3rd Qu.: 29.50   3rd Qu.: 8.000
##                                         Max.   :150.00   Max.   :15.000
##                                         NA's   :15       NA's   :20
##     notas
##  Length:158
##  Class :character
##  Mode  :character
##
##
##
##
```

```r
dim(df)           # Dimensiones (filas, columnas)
```

```
## [1] 158  13
```

```r
names(df)         # Nombres de las columnas
```

```
##  [1] "id"            "nombre"        "edad"          "salario_texto"
##  [5] "departamento"  "fecha_ingreso" "email"         "telefono"
##  [9] "genero"        "estado_civil"  "experiencia"   "puntaje"
## [13] "notas"
```

### 3.2.2 1.2 Identificación de Problemas

```r
# Verificar valores únicos
sapply(df, function(x) length(unique(x)))
```

```
##            id        nombre          edad salario_texto  departamento
##           141            13            50           149            14
## fecha_ingreso         email      telefono        genero  estado_civil
##           126           135           145             9             5
```

```
##   experiencia       puntaje         notas
##            42            14             8
```

```
# Identificar tipos de datos incorrectos
sapply(df, class)
```

```
##            id        nombre          edad salario_texto  departamento
##     "integer"   "character"     "numeric"   "character"   "character"
## fecha_ingreso         email      telefono        genero  estado_civil
##   "character"   "character"   "character"   "character"   "character"
##   experiencia       puntaje         notas
##     "numeric"     "numeric"   "character"
```

```
# Detectar patrones anómalos en variables categóricas
table(df$departamento, useNA = "ifany")
```

```
##
##         finanzas         Finanzas               IT        Marketing
##                8               12               12               14
##        MARKETING Recursos Humanos             rrhh             RRHH
##               12                8               12               18
##         sistemas         Sistemas           ventas           Ventas
##                9               14               12                9
##           VENTAS             <NA>
##                7               11
```

```
table(df$genero, useNA = "ifany")
```

```
##
##         f         F  Femenino         H         m         M Masculino     Mujer
##        16        15        11        22        16        13        22        21
##      <NA>
##        22
```

```
table(df$estado_civil, useNA = "ifany")
```

```
##
##     Casado Divorciado    Soltero      Viudo       <NA>
##         38         35         36         24         25
```

## 3.3 2. Manejo de Valores Faltantes (NA)

### 3.3.1 2.1 Detección de Valores Faltantes

```
# Contar NA por columna
colSums(is.na(df))
```

```
##            id        nombre          edad salario_texto  departamento
##             3            18            18             5            11
## fecha_ingreso         email      telefono        genero  estado_civil
##            15            19             9            22            25
##   experiencia       puntaje         notas
##            15            20            13
```

```
# Porcentaje de NA por columna
colMeans(is.na(df)) * 100
```

```
##            id        nombre          edad salario_texto  departamento
```

```
##      1.898734     11.392405     11.392405      3.164557      6.962025
## fecha_ingreso         email      telefono        genero  estado_civil
##      9.493671     12.025316      5.696203     13.924051     15.822785
##   experiencia       puntaje         notas
##      9.493671     12.658228      8.227848
```

```
# Visualizar patrones de NA
sum(is.na(df))                    # Total de NA
```

```
## [1] 193
```

```
sum(complete.cases(df))           # Filas completas
```

```
## [1] 94
```

```
nrow(df) - sum(complete.cases(df)) # Filas con al menos un NA
```

```
## [1] 64
```

### 3.3.2 2.2 Estrategias para Manejar NA

```
# Eliminar filas con cualquier NA
df_clean <- na.omit(df)
cat("Filas antes:", nrow(df), "- Filas después:", nrow(df_clean), "\n")
```

#### 3.3.2.1 Eliminación de NA

```
## Filas antes: 158 - Filas después: 94
```

```
# Eliminar filas con NA en columnas específicas importantes
df_filtrado <- df[complete.cases(df[, c("id", "edad")]), ]
cat("Filas después de filtrar por ID y edad:", nrow(df_filtrado), "\n")
```

```
## Filas después de filtrar por ID y edad: 140
```

```
# Crear copia para trabajar
df_working <- df

# Reemplazar con media (variables numéricas)
df_working$edad[is.na(df_working$edad)] <- mean(df_working$edad, na.rm = TRUE)
df_working$experiencia[is.na(df_working$experiencia)] <- mean(df_working$experiencia, na.rm = TRUE)
df_working$puntaje[is.na(df_working$puntaje)] <- mean(df_working$puntaje, na.rm = TRUE)

# Reemplazar con moda (variables categóricas)
# Función para calcular moda
calcular_moda <- function(x) {
  x <- x[!is.na(x) & x != ""]
  if(length(x) == 0) return(NA)
  tabla <- table(x)
  names(tabla)[which.max(tabla)]
}

moda_departamento <- calcular_moda(df_working$departamento)
df_working$departamento[is.na(df_working$departamento)] <- moda_departamento

moda_genero <- calcular_moda(df_working$genero)
```

```r
df_working$genero[is.na(df_working$genero)] <- moda_genero

moda_estado_civil <- calcular_moda(df_working$estado_civil)
df_working$estado_civil[is.na(df_working$estado_civil)] <- moda_estado_civil

# Verificar resultado
colSums(is.na(df_working))
```

#### 3.3.2.2 Reemplazo de NA

```
##            id        nombre          edad salario_texto  departamento
##             3            18             0             5             0
## fecha_ingreso         email      telefono        genero  estado_civil
##            15            19             9             0             0
##   experiencia       puntaje         notas
##             0             0            13
```

## 3.4 3. Detección y Manejo de Valores Atípicos

### 3.4.1 3.1 Identificación de Outliers

```r
# Método del rango intercuartílico (IQR) para edad
Q1_edad <- quantile(df_working$edad, 0.25, na.rm = TRUE)
Q3_edad <- quantile(df_working$edad, 0.75, na.rm = TRUE)
IQR_edad <- Q3_edad - Q1_edad

# Límites para outliers en edad
limite_inferior_edad <- Q1_edad - 1.5 * IQR_edad
limite_superior_edad <- Q3_edad + 1.5 * IQR_edad

# Identificar outliers en edad
outliers_edad <- df_working$edad < limite_inferior_edad |
                 df_working$edad > limite_superior_edad |
                 df_working$edad < 0 |
                 df_working$edad > 100

cat("Outliers en edad:", sum(outliers_edad, na.rm = TRUE), "\n")
```

```
## Outliers en edad: 3
```

```r
cat("Valores problemáticos en edad:\n")
```

```
## Valores problemáticos en edad:
```

```r
print(df_working$edad[outliers_edad])
```

```
## [1] 150  -5 999
```

```r
# Outliers en experiencia (valores negativos o muy altos)
outliers_exp <- df_working$experiencia < 0 | df_working$experiencia > 50
cat("Outliers en experiencia:", sum(outliers_exp, na.rm = TRUE), "\n")
```

```
## Outliers en experiencia: 3
```

```r
# Outliers en puntaje (fuera del rango 1-10)
outliers_puntaje <- df_working$puntaje < 1 | df_working$puntaje > 10
cat("Outliers en puntaje:", sum(outliers_puntaje, na.rm = TRUE), "\n")
```

```
## Outliers en puntaje: 3
```

#### 3.4.2 3.2 Tratamiento de Outliers

```
# Eliminar outliers extremos en edad
df_working <- df_working[!outliers_edad, ]

# Corregir experiencia negativa (convertir a 0)
df_working$experiencia[df_working$experiencia < 0] <- 0

# Corregir experiencia muy alta (limitar a 40)
df_working$experiencia[df_working$experiencia > 40] <- 40

# Corregir puntajes fuera de rango
df_working$puntaje[df_working$puntaje < 1] <- 1
df_working$puntaje[df_working$puntaje > 10] <- 10

cat("Filas después de limpiar outliers:", nrow(df_working), "\n")
```

```
## Filas después de limpiar outliers: 155
```

### 3.5 4. Normalización de Datos

#### 3.5.1 4.1 Conversión de Salarios

```
# El salario está como texto, necesitamos convertirlo a numérico
# Primero veamos algunos ejemplos
head(df_working$salario_texto, 10)
```

```
##  [1] "$ 35078" "$68209"  "$78433"  "$ 67101" "$72220"  "$36311"  "$40832"
##  [8] "$33246"  "$47048"  "$67768"
```

```
# Limpiar y convertir salarios
df_working$salario_numerico <- df_working$salario_texto %>%
  gsub("\\$", "", .) %>%          # Quitar signo $
  gsub(",", "", .) %>%            # Quitar comas
  gsub(" ", "", .) %>%            # Quitar espacios
  gsub("\\.00$", "", .) %>%       # Quitar .00 al final
  as.numeric()

# Ver resultados
summary(df_working$salario_numerico)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       5   40630   53042   70168   69320 1500000       5
```

```
# Manejar outliers en salario (muy altos o muy bajos)
Q1_sal <- quantile(df_working$salario_numerico, 0.25, na.rm = TRUE)
Q3_sal <- quantile(df_working$salario_numerico, 0.75, na.rm = TRUE)
IQR_sal <- Q3_sal - Q1_sal

# Identificar outliers extremos
outliers_sal <- df_working$salario_numerico < 10000 |
                df_working$salario_numerico > 200000

cat("Outliers extremos en salario:", sum(outliers_sal, na.rm = TRUE), "\n")
```

```
## Outliers extremos en salario: 3
```

```
# Filtrar outliers extremos de salario
df_working <- df_working[!outliers_sal | is.na(outliers_sal), ]
```

### 3.5.2 4.2 Formato y Consistencia

```
# Estandarizar nombres (eliminar espacios extra y capitalizar)
df_working <- df_working %>%
  mutate(
    # Estandarizar nombres (eliminar espacios extra y capitalizar)
    nombre_limpio = nombre %>%
      trimws() %>%                    # Eliminar espacios al inicio y final
      gsub("\\s+", " ", .) %>%        # Reemplazar múltiples espacios con uno solo
      tools::toTitleCase(.),          # Formato título

    # Estandarizar departamentos - FIXED: wrapped case_when in {}
    departamento_limpio = {
      temp_dept <- departamento %>% trimws() %>% tolower()
      case_when(
        temp_dept %in% c("ventas", "venta") ~ "Ventas",
        temp_dept %in% c("marketing") ~ "Marketing",
        temp_dept %in% c("recursos humanos", "rrhh") ~ "Recursos Humanos",
        temp_dept %in% c("sistemas", "it") ~ "Sistemas",
        temp_dept %in% c("finanzas") ~ "Finanzas",
        TRUE ~ tools::toTitleCase(temp_dept)
      )
    },

    # Estandarizar género - FIXED: avoid piping directly into case_when
    genero_limpio = case_when(
      toupper(genero) %in% c("M", "MASCULINO", "H") ~ "Masculino",
      toupper(genero) %in% c("F", "FEMENINO", "MUJER") ~ "Femenino",
      TRUE ~ genero
    )
  )

# Ver resultados
table(df_working$departamento_limpio, useNA = "ifany")
```

```
##
##         Finanzas        Marketing Recursos Humanos         Sistemas
##               20               25               47               34
##           Ventas
##               26
```

```
table(df_working$genero_limpio, useNA = "ifany")
```

```
##
##  Femenino Masculino
##        62        90
```

### 3.5.3 4.3 Conversión de Fechas

```
# Convertir fechas de ingreso
# Primero veamos el formato actual
head(df_working$fecha_ingreso, 10)
```

```
##  [1] NA           "16/05/2020" "24/04/2021" ""           "16/07/2020"
##  [6] "19/01/2023" "22/11/2020" "22/04/2017" "09/02/2018" "20/11/2021"
```

```
# Convertir fechas válidas
df_working$fecha_ingreso_date <- as.Date(df_working$fecha_ingreso, format = "%d/%m/%Y")

# Identificar fechas problemáticas (futuras o muy antiguas)
fechas_problema <- df_working$fecha_ingreso_date > Sys.Date() |
                   df_working$fecha_ingreso_date < as.Date("1990-01-01")

cat("Fechas problemáticas:", sum(fechas_problema, na.rm = TRUE), "\n")
```

```
## Fechas problemáticas: 0
```

```
# Filtrar fechas problemáticas
df_working$fecha_ingreso_date[fechas_problema] <- NA

# Ver resultado
summary(df_working$fecha_ingreso_date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2015-05-19" "2017-12-05" "2019-04-19" "2019-07-01" "2021-04-27" "2023-08-31"
##         NA's
##         "22"
```

## 3.6 5. Validación de Datos

### 3.6.1 5.1 Verificación de Rangos

```
# Verificar rangos lógicos en edad
cat("Rango de edades:", range(df_working$edad, na.rm = TRUE), "\n")
```

```
## Rango de edades: 2 65
```

```
# Verificar que no hay salarios negativos o extremos
cat("Rango de salarios:", range(df_working$salario_numerico, na.rm = TRUE), "\n")
```

```
## Rango de salarios: 26497 85586
```

```
# Verificar experiencia
cat("Rango de experiencia:", range(df_working$experiencia, na.rm = TRUE), "\n")
```

```
## Rango de experiencia: 0 40
```

```
# Verificar puntajes
cat("Rango de puntajes:", range(df_working$puntaje, na.rm = TRUE), "\n")
```

```
## Rango de puntajes: 1 10
```

```
# Mostrar casos problemáticos si los hay
problemas_edad <- df_working[df_working$edad < 18 | df_working$edad > 65, ]
if(nrow(problemas_edad) > 0) {
  cat("Casos con edades fuera del rango laboral típico:\n")
```

```
  print(problemas_edad[, c("id", "edad", "experiencia")])
}
```

```
## Casos con edades fuera del rango laboral típico:
##      id edad experiencia
## 38  134   15          37
## 68  135    2          27
## 78  132    5          36
## 90  130    9          20
## 93  133   15          24
## 101 131    9           9
## 114 129    5           2
```

### 3.6.2 5.2 Consistencia entre Variables

```
# Verificar consistencia entre edad y experiencia
# La experiencia no debería ser mayor que edad - 16
inconsistencia <- df_working$experiencia > (df_working$edad - 16)
cat("Casos con experiencia inconsistente:", sum(inconsistencia, na.rm = TRUE), "\n")
```

```
## Casos con experiencia inconsistente: 62
```

```
if(sum(inconsistencia, na.rm = TRUE) > 0) {
  cat("Casos problemáticos:\n")
  print(df_working[inconsistencia, c("id", "edad", "experiencia")])

  # Corregir ajustando experiencia
  df_working$experiencia[inconsistencia] <- pmax(0, df_working$edad[inconsistencia] - 18)
}
```

```
## Casos problemáticos:
##      id     edad experiencia
## 2   116 19.00000          4
## 7    31 42.00000         28
## 9   119 20.00000         25
## 12   24 31.00000         31
## 19   12 38.00000         37
## 20   33 35.00000         40
## 21    2 20.00000         20
## 22   36 24.00000          9
## 26  123 29.00000         27
## 27   72 38.00000         25
## 33   63 23.00000         25
## 36  101 23.00000         10
## 37   90 27.00000         30
## 38  134 15.00000         37
## 42   73 24.00000         36
## 48    2 20.00000         20
## 58   55 37.00000         26
## 59   96 38.00000         39
## 61   27 50.00000         39
## 65   32 25.00000         40
## 68  135  2.00000         27
## 70   51 42.00000         40
## 72   38 41.00000         34
```

```
## 73   34 26.00000          36
## 74   18 23.00000          34
## 76  105 34.00000          23
## 78  132  5.00000          36
## 80   94 25.00000          27
## 81   83 37.00000          27
## 83   44 27.00000          14
## 86   76 37.00000          28
## 88   89 21.00000          15
## 90  130  9.00000          20
## 93  133 15.00000          24
## 95  138 46.56429          40
## 97  120 35.00000          27
## 99   65 18.00000          11
## 101 131  9.00000           9
## 104  88 19.00000          21
## 105  22 53.00000          40
## 108 115 19.00000          23
## 110  57 26.00000          26
## 111  10 25.00000          20
## 113 118 41.00000          40
## 114 129  5.00000           2
## 116  74 43.00000          30
## 117 107 45.00000          33
## 119  56 41.00000          33
## 121  45 30.00000          33
## 129  92 22.00000          19
## 133  14 19.00000          35
## 134   8 37.00000          38
## 136 102 24.00000          40
## 138 137 46.56429          40
## 144 113 34.00000          28
## 145 112 46.00000          38
## 146  16 30.00000          36
## 149  77 23.00000          23
## 150 121 19.00000           5
## 151  39 40.00000          38
## 152   9 28.00000          29
## 154  81 34.00000          34
```

```r
# Verificar que las fechas de ingreso sean coherentes con la experiencia
df_working$años_desde_ingreso <- as.numeric(Sys.Date() - df_working$fecha_ingreso_date) / 365.25
diferencia_exp <- abs(df_working$experiencia - df_working$años_desde_ingreso)
casos_inconsistentes <- diferencia_exp > 5 & !is.na(diferencia_exp)

cat("Casos con experiencia y fecha de ingreso inconsistentes:", sum(casos_inconsistentes, na.rm = TRUE))
```

```
## Casos con experiencia y fecha de ingreso inconsistentes: 72
```

## 3.7 6. Eliminación de Duplicados

### 3.7.1 6.1 Identificación de Duplicados

```r
# Filas completamente duplicadas
filas_duplicadas <- sum(duplicated(df_working))
```

```r
cat("Filas completamente duplicadas:", filas_duplicadas, "\n")
```

```
## Filas completamente duplicadas: 7
```

```r
# Duplicados por ID
ids_duplicados <- sum(duplicated(df_working$id, incomparables = NA))
cat("IDs duplicados:", ids_duplicados, "\n")
```

```
## IDs duplicados: 12
```

```r
# Ver los duplicados por ID
if(ids_duplicados > 0) {
  duplicados_id <- df_working[df_working$id %in% df_working$id[duplicated(df_working$id)], ]
  print(duplicados_id[order(duplicados_id$id), c("id", "nombre_limpio", "edad")])
}
```

```
##     id    nombre_limpio     edad
## 30   1     SOFIA TORRES 57.00000
## 153  1     SOFIA TORRES 57.00000
## 21   2 Carmen Rodríguez 20.00000
## 48   2 Carmen Rodríguez 20.00000
## 96   2             <NA> 46.56429
## 77   3        Ana López 46.00000
## 79   3        Ana López 46.00000
## 84   3             <NA> 46.56429
## 132  3             <NA> 46.56429
## 69   4     SOFIA TORRES 53.00000
## 143  4     SOFIA TORRES 53.00000
## 11   5        ANA MARÍA 61.00000
## 109  5        ANA MARÍA 61.00000
## 56   6        ANA MARÍA 39.00000
## 126  6             <NA> 46.56429
## 102  9                  46.56429
## 152  9      Luis Miguel 28.00000
## 4   10             <NA> 46.56429
## 100 10                  46.56429
## 111 10      Carlos Ruiz 25.00000
## 67  NA             <NA> 46.56429
## 112 NA             <NA> 46.56429
## 142 NA             <NA> 46.56429
```

```r
# Duplicados en nombre (posibles personas repetidas)
duplicados_nombre <- df_working[duplicated(df_working$nombre_limpio) |
                                duplicated(df_working$nombre_limpio, fromLast = TRUE), ]
if(nrow(duplicados_nombre) > 0) {
  cat("Posibles nombres duplicados:", nrow(duplicados_nombre), "\n")
}
```

```
## Posibles nombres duplicados: 152
```

### 3.7.2 6.2 Eliminación de Duplicados

```r
# Eliminar filas completamente duplicadas
df_working <- df_working[!duplicated(df_working), ]

# Para IDs duplicados, mantener el registro más completo
```

```r
# (el que tenga menos NA)
df_working <- df_working %>%
  group_by(id) %>%
  # Calculate NA count for each row within each group
  mutate(na_count = rowSums(is.na(across(everything())))) %>%
  slice_min(na_count, n = 1, with_ties = FALSE) %>%
  select(-na_count) %>%  # Remove the helper column
  ungroup()

cat("Filas después de eliminar duplicados:", nrow(df_working), "\n")
```

```
## Filas después de eliminar duplicados: 138
```

## 3.8 7. Restructuración de Datos

### 3.8.1 7.1 Seleccionar y Renombrar Variables Finales

```r
# Seleccionar las columnas limpias y renombrarlas
df_final <- df_working %>%
  select(
    id = id,
    nombre = nombre_limpio,
    edad = edad,
    salario = salario_numerico,
    departamento = departamento_limpio,
    fecha_ingreso = fecha_ingreso,
    email = email,
    telefono = telefono,
    genero = genero_limpio,
    estado_civil = estado_civil,
    experiencia = experiencia,
    puntaje = puntaje,
    notas = notas
  ) %>%
  # Filtrar solo registros con información básica válida
  filter(!is.na(id), !is.na(edad), !is.na(salario))

# Mostrar estructura final
str(df_final)
```

```
## tibble [137 x 13] (S3: tbl_df/tbl/data.frame)
##  $ id           : int [1:137] 1 2 3 4 5 6 7 8 9 10 ...
##  $ nombre       : chr [1:137] "SOFIA TORRES" "Carmen Rodríguez" "Ana López" "SOFIA TORRES" ...
##  $ edad         : num [1:137] 57 20 46 53 61 39 59 37 28 25 ...
##  $ salario      : num [1:137] 49418 45702 36622 38982 67768 ...
##  $ departamento : chr [1:137] "Recursos Humanos" "Ventas" "Recursos Humanos" "Recursos Humanos" ...
##  $ fecha_ingreso: chr [1:137] "14/08/2018" "16/01/2021" "30/08/2019" "02/04/2019" ...
##  $ email        : chr [1:137] "maria754@empresa.com" "maria835@empresa.com" "maria385@empresa.com" "m
##  $ telefono     : chr [1:137] "011-8132-8730" "011-4754-8834" "011-8133-3634" "011-7257-9624" ...
##  $ genero       : chr [1:137] "Masculino" "Masculino" "Masculino" "Femenino" ...
##  $ estado_civil : chr [1:137] "Divorciado" "Soltero" "Casado" "Casado" ...
##  $ experiencia  : num [1:137] 38 2 3 9 15 15 34 19 10 7 ...
##  $ puntaje      : num [1:137] 1 3 8 2 3 10 1 7 3 5 ...
##  $ notas        : chr [1:137] "  Buen desempeño  " "Sin comentarios" "NECESITA MEJORAS" "Sin comentar
```

### 3.8.2 7.2 Crear Variables Derivadas

```
df_final <- df_final %>%
  mutate(
    # Crear categorías de edad
    grupo_edad = case_when(
      edad < 30 ~ "Joven",
      edad < 50 ~ "Adulto",
      TRUE ~ "Mayor"
    ),

    # Categorizar salario
    categoria_salario = case_when(
      salario < 40000 ~ "Bajo",
      salario < 70000 ~ "Medio",
      TRUE ~ "Alto"
    ),

    # Calcular años desde ingreso - FIXED: handle non-date values
    años_empresa = case_when(
      is.na(fecha_ingreso) ~ NA_real_,
      !is.na(fecha_ingreso) ~ as.numeric(Sys.Date() - as.Date(fecha_ingreso)) / 365.25,
      TRUE ~ NA_real_
    ),

    # Categorizar experiencia
    nivel_experiencia = case_when(
      experiencia < 5 ~ "Junior",
      experiencia < 15 ~ "Semi-Senior",
      TRUE ~ "Senior"
    ),

    # Crear score combinado - FIXED: handle NA values in años_empresa
    score_empleado = (puntaje * 0.6) +
                     (pmin(experiencia, 20, na.rm = TRUE) * 0.2) +
                     (pmin(coalesce(años_empresa, 0), 10) * 0.2)
  )

# Ver las nuevas variables
table(df_final$grupo_edad)
```

```
##
## Adulto  Joven  Mayor
##     61     40     36
```

```
table(df_final$categoria_salario)
```

```
##
##  Alto  Bajo Medio
##    35    32    70
```

```
table(df_final$nivel_experiencia)
```

```
##
##      Junior Semi-Senior      Senior
##          36          50          51
```

```r
# Check the años_empresa calculation
summary(df_final$años_empresa)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1994    2001    2008    2009    2018    2024      12
```

### 3.9 8. Uso de dplyr para Limpieza Eficiente

```r
library(dplyr)

df_clean_dplyr <- df %>%
  # Filtrar datos válidos básicos
  filter(!is.na(id),
         !is.na(edad), edad > 17, edad < 70,
         !is.na(salario_texto)) %>%

  # Limpiar y transformar datos
  mutate(
    # Limpiar nombres
    nombre_clean = trimws(nombre) %>%
      gsub("\\s+", " ", .) %>%
      tools::toTitleCase(.),

    # Convertir salario
    salario_clean = salario_texto %>%
      gsub("[\\$,\\s]", "", .) %>%
      gsub("\\.00$", "", .) %>%
      as.numeric(),

    # Estandarizar departamento
    depto_clean = case_when(
      tolower(trimws(departamento)) %in% c("ventas", "venta") ~ "Ventas",
      tolower(trimws(departamento)) %in% c("marketing") ~ "Marketing",
      tolower(trimws(departamento)) %in% c("recursos humanos", "rrhh") ~ "RRHH",
      tolower(trimws(departamento)) %in% c("sistemas", "it") ~ "IT",
      tolower(trimws(departamento)) %in% c("finanzas") ~ "Finanzas",
      TRUE ~ "Otros"
    ),

    # Limpiar género
    genero_clean = case_when(
      toupper(genero) %in% c("M", "MASCULINO", "H") ~ "M",
      toupper(genero) %in% c("F", "FEMENINO", "MUJER") ~ "F",
      TRUE ~ "Otro"
    )
  ) %>%

  # Filtrar outliers de salario
  filter(salario_clean >= 20000, salario_clean <= 150000) %>%

  # Seleccionar columnas relevantes
  select(id, nombre_clean, edad, salario_clean, depto_clean,
         genero_clean, experiencia, puntaje) %>%
```

```
  # Eliminar duplicados
  distinct() %>%

  # Ordenar
  arrange(id)

cat("Dataset final con dplyr:", nrow(df_clean_dplyr), "filas\n")
```

```
## Dataset final con dplyr: 125 filas
```

## 3.10 9. Validación Final

### 3.10.1 9.1 Verificaciones Post-Limpieza

```
# Verificar estructura final
str(df_final)
```

```
## tibble [137 x 18] (S3: tbl_df/tbl/data.frame)
##  $ id               : int [1:137] 1 2 3 4 5 6 7 8 9 10 ...
##  $ nombre           : chr [1:137] "SOFIA TORRES" "Carmen Rodríguez" "Ana López" "SOFIA TORRES" ...
##  $ edad             : num [1:137] 57 20 46 53 61 39 59 37 28 25 ...
##  $ salario          : num [1:137] 49418 45702 36622 38982 67768 ...
##  $ departamento     : chr [1:137] "Recursos Humanos" "Ventas" "Recursos Humanos" "Recursos Humanos"
##  $ fecha_ingreso    : chr [1:137] "14/08/2018" "16/01/2021" "30/08/2019" "02/04/2019" ...
##  $ email            : chr [1:137] "maria754@empresa.com" "maria835@empresa.com" "maria385@empresa.co
##  $ telefono         : chr [1:137] "011-8132-8730" "011-4754-8834" "011-8133-3634" "011-7257-9624" ..
##  $ genero           : chr [1:137] "Masculino" "Masculino" "Masculino" "Femenino" ...
##  $ estado_civil     : chr [1:137] "Divorciado" "Soltero" "Casado" "Casado" ...
##  $ experiencia      : num [1:137] 38 2 3 9 15 15 34 19 10 7 ...
##  $ puntaje          : num [1:137] 1 3 8 2 3 10 1 7 3 5 ...
##  $ notas            : chr [1:137] "  Buen desempeño  " "Sin comentarios" "NECESITA MEJORAS" "Sin com
##  $ grupo_edad       : chr [1:137] "Mayor" "Joven" "Adulto" "Mayor" ...
##  $ categoria_salario: chr [1:137] "Medio" "Medio" "Bajo" "Bajo" ...
##  $ años_empresa     : num [1:137] 2011 2010 1995 2023 2005 ...
##  $ nivel_experiencia: chr [1:137] "Senior" "Junior" "Junior" "Semi-Senior" ...
##  $ score_empleado   : num [1:137] 6.6 4.2 7.4 5 6.8 11 6.6 10 5.8 6.4 ...
```

```
# Verificar que no hay NA en campos críticos
cat("NA en campos críticos:\n")
```

```
## NA en campos críticos:
```

```
cat("ID:", sum(is.na(df_final$id)), "\n")
```

```
## ID: 0
```

```
cat("Edad:", sum(is.na(df_final$edad)), "\n")
```

```
## Edad: 0
```

```
cat("Salario:", sum(is.na(df_final$salario)), "\n")
```

```
## Salario: 0
```

```
# Verificar rangos finales
cat("\nRangos finales:\n")
```

```
##
```

```
## Rangos finales:
```

```
cat("Edad:", range(df_final$edad, na.rm = TRUE), "\n")
```

```
## Edad: 2 65
```

```
cat("Salario:", range(df_final$salario, na.rm = TRUE), "\n")
```

```
## Salario: 26497 85586
```

```
cat("Experiencia:", range(df_final$experiencia, na.rm = TRUE), "\n")
```

```
## Experiencia: 0 39
```

```
cat("Puntaje:", range(df_final$puntaje, na.rm = TRUE), "\n")
```

```
## Puntaje: 1 10
```

```
# Verificar categorías
cat("\nDistribución por categorías:\n")
```

```
##
## Distribución por categorías:
```

```
table(df_final$grupo_edad)
```

```
##
## Adulto  Joven  Mayor
##     61     40     36
```

```
table(df_final$departamento)
```

```
##
##         Finanzas        Marketing Recursos Humanos         Sistemas
##               16               25               39               32
##           Ventas
##               25
```

```
table(df_final$genero)
```

```
##
##  Femenino Masculino
##        59        78
```

```
# Comparar antes y después
cat("\nComparación antes y después:\n")
```

```
##
## Comparación antes y después:
```

```
cat("Filas originales:", nrow(df), "\n")
```

```
## Filas originales: 158
```

```
cat("Filas después de limpieza:", nrow(df_final), "\n")
```

```
## Filas después de limpieza: 137
```

```
cat("Porcentaje retenido:", round(nrow(df_final)/nrow(df)*100, 2), "%\n")
```

```
## Porcentaje retenido: 86.71 %
```

```r
# Resumen de calidad
cat("\nCalidad final del dataset:\n")
```

```
##
## Calidad final del dataset:
```

```r
cat("Total de celdas:", nrow(df_final) * ncol(df_final), "\n")
```

```
## Total de celdas: 2466
```

```r
cat("Celdas con NA:", sum(is.na(df_final)), "\n")
```

```
## Celdas con NA: 40
```

```r
cat("Porcentaje completitud:", round((1 - sum(is.na(df_final))/(nrow(df_final) * ncol(df_final)))*100, 1
```

```
## Porcentaje completitud: 98.38 %
```

## 3.11 10. Mejores Prácticas

### 3.11.1 10.1 Documentación del Proceso

```r
# Mantener registro de cambios
limpieza_log <- data.frame(
  paso = c("Datos originales",
           "Eliminar outliers edad",
           "Convertir salarios",
           "Estandarizar categorías",
           "Eliminar duplicados",
           "Dataset final"),
  filas = c(nrow(df),
            nrow(df_working),
            nrow(df_working),
            nrow(df_working),
            nrow(df_working),
            nrow(df_final)),
  columnas = c(ncol(df), ncol(df_working), ncol(df_working) + 1,
               ncol(df_working) + 3, ncol(df_working) + 3, ncol(df_final)),
  completitud = c(round((1 - sum(is.na(df))/(nrow(df) * ncol(df)))*100, 1),
                  round((1 - sum(is.na(df_working))/(nrow(df_working) * ncol(df_working)))*100, 1),
                  round((1 - sum(is.na(df_working))/(nrow(df_working) * ncol(df_working)))*100, 1),
                  round((1 - sum(is.na(df_working))/(nrow(df_working) * ncol(df_working)))*100, 1),
                  round((1 - sum(is.na(df_working))/(nrow(df_working) * ncol(df_working)))*100, 1),
                  round((1 - sum(is.na(df_final))/(nrow(df_final) * ncol(df_final)))*100, 1))
)

print(limpieza_log)
```

```
##                      paso filas columnas completitud
## 1        Datos originales   158       13        90.6
## 2  Eliminar outliers edad   138       19        97.2
## 3      Convertir salarios   138       20        97.2
## 4 Estandarizar categorías   138       22        97.2
## 5     Eliminar duplicados   138       22        97.2
## 6           Dataset final   137       18        98.4
```

### 3.11.2 10.2 Funciones Reutilizables

```r
# Función para limpieza básica del dataset generado
limpiar_dataset_empleados <- function(df) {
  df %>%
    # Filtrar registros válidos básicos
    filter(!is.na(id), !is.na(edad), edad > 0, edad < 100) %>%

    # Limpiar texto
    mutate(across(where(is.character), ~trimws(.))) %>%

    # Convertir salarios
    mutate(salario_numerico = salario_texto %>%
             gsub("[\\$,\\s]", "", .) %>%
             gsub("\\.00$", "", .) %>%
             as.numeric()) %>%

    # Filtrar salarios válidos
    filter(salario_numerico > 0, salario_numerico < 500000) %>%

    # Eliminar duplicados
    distinct(id, .keep_all = TRUE) %>%

    # Ordenar
    arrange(id)
}

# Función para detectar outliers mejorada
detectar_outliers_mejorado <- function(x, factor = 1.5) {
  Q1 <- quantile(x, 0.25, na.rm = TRUE)
  Q3 <- quantile(x, 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1
  limite_inf <- Q1 - factor * IQR
  limite_sup <- Q3 + factor * IQR

  list(
    outliers = x < limite_inf | x > limite_sup,
    limite_inferior = limite_inf,
    limite_superior = limite_sup,
    cantidad = sum(x < limite_inf | x > limite_sup, na.rm = TRUE)
  )
}

# Probar la función de limpieza
df_prueba <- limpiar_dataset_empleados(df)
cat("Resultado función de limpieza:", nrow(df_prueba), "filas\n")
```

```
## Resultado función de limpieza: 132 filas
```

```r
# Probar función de outliers
outliers_edad <- detectar_outliers_mejorado(df_final$edad)
cat("Outliers detectados en edad:", outliers_edad$cantidad, "\n")
```

```
## Outliers detectados en edad: 0
```

## 3.12 Conclusión

La limpieza de datos es un proceso iterativo que requiere:

- Comprensión profunda de los datos y su contexto
- Aplicación sistemática de técnicas de validación

- Documentación clara de todos los cambios realizados
- Verificación constante de la calidad de los datos

Un dataset bien limpio es la base para análisis confiables y resultados reproducibles. El tiempo invertido en esta etapa se traduce en mayor precisión y confiabilidad en los análisis posteriores.

```r
# Exportar dataset limpio final
write.csv(df_final, "dataset_empleados_limpio.csv", row.names = FALSE)
# cat("Dataset limpio exportado como: dataset_empleados_limpio.csv\n")
```