Metadata are an essential part of a robust data science workflow ; they record the meaning of each variable : its units, quality, allowed range, how we collect it, when it’s been recorded etc. Data without metadata are practically worthless. Here we will show how to transfer the metadata from PostgreSQL to R.

In PostgreSQL metadata can be stored in comments with the [statements](https://www.postgresql.org/docs/current/sql-comment.html) COMMENT ON COLUMN... or COMMENT ON TABLE.... So I hope your tables have this nice comments, and you can see them in PgAdmin for example. But what about R ?

In R metadata can be assigned as attributes of any object and mainly as « labels ». We will use the Hmisc package which provides functions to manage labels. Another interesting package is sjlabelled. You may also have seen labels when importing labelled data from SPSS for example.

library(Hmisc)

library(tidyverse)

library(RPostgreSQL)

library(glue)

library(sjlabelled)

cnx <- dbConnect(dbDriver("PostgreSQL"),

user = "\*\*\*",

password = "\*\*\*",

host = "\*\*\*",

dbname = "\*\*\*",

port = 5432

)

Usually we got some data this way :

cog <- dbGetQuery(cnx,

"SELECT

\*

FROM ref\_cog.france2018

LIMIT 10")

We can create a function that will query the metadata of the table in information\_schema.columns and add it to the data frame ; the function expects a data frame, the name of the schema.table from which we get the comments and a connection handler. It will return the data frame with labels :

#' Add attributes to a dataframe from metadata read in the PostgreSQL database

#'

#' @param df dataframe

#' @param schema\_table "schema.table" from which to read the comments

#' @param cnx a database connexion from RPostgreSQL::dbConnect()

#'

#' @return a dataframe with attributes

#'

#' @examples \dontrun{add\_metadata(iris, "public.iris", cnx)}

add\_metadata <- function(df, schema\_table, cnx) {

meta <- str\_match(schema\_table, "^(.\*)\\.(.\*)$") %>%

glue\_sql(

"SELECT

column\_name,

pg\_catalog.col\_description(

format('%s.%s', isc.table\_schema, isc.table\_name)::regclass::oid,

isc.ordinal\_position) AS column\_description

FROM information\_schema.columns AS isc

WHERE isc.table\_schema = {s[2]}

AND isc.table\_name = {s[3]};",

s = .,

.con = cnx) %>%

dbGetQuery(cnx, .)

label(df, self = FALSE) <- colnames(df) %>%

enframe() %>%

left\_join(meta, by = c("value" = "column\_name")) %>%

pull(column\_description)

df

}

Now we would do :

cog <- dbGetQuery(cnx,

"SELECT

\*

FROM ref\_cog.france2018

LIMIT 10") %>%

add\_metadata("ref\_cog.france2018", cnx)

You can see the metadata in the column headings of the RStudio viewer with View(cog) or with contents(cog). Or with :

cog %>%

label() %>%

enframe()

Or lastly, for one column :

label(cog$cheflieu)

We can also search for information in the variable names or in the labels with another function :

search\_var <- function(df, keyword) {

df %>%

label() %>%

enframe() %>%

rename(variable = name,

metadata = value) %>%

filter\_all(any\_vars(str\_detect(., regex(keyword, ignore\_case = TRUE))))

}

search\_var(cog, "canton")