Etapa 1. Conociendo el negocio

2025-08-21

```
library(readxl)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyr)
library(stringr)
library(lubridate)

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(janitor)

##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test

library(purrr)

## Warning: package 'purrr' was built under R version 4.5.1

library(openxlsx)
library(zoo)

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library(imputeTS)

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

##
## Attaching package: 'imputeTS'

## The following object is masked from 'package:zoo':
##
##     na.locf

# --- helpers simples ---
mask_short_na_runs <- function(x, maxlen) {
  nas <- is.na(x); r <- rle(nas)
  ends <- cumsum(r$lengths); starts <- ends - r$lengths + 1
  mask <- rep(FALSE, length(x))
  for (i in seq_along(r$lengths)) {
    if (r$values[i]) {
      if (r$lengths[i] <= maxlen) mask[starts[i]:ends[i]] <- TRUE
    }
  }
  mask
}

safe_kalman_struct <- function(x) {
  out <- tryCatch(
    suppressWarnings(na_kalman(x, model = "StructTS")),
    warning = function(w) x,
    error   = function(e) x
  )
  out
}

fill_two_stage <- function(v, maxgap_linear = 6, maxgap_kalman = 24) {
  # 1) lineal
  out <- na.approx(v, maxgap = maxgap_linear, na.rm = FALSE)
  # 2) kalman solo en huecos cortos restantes
  if (maxgap_kalman > 0) {
    still_na <- is.na(out)
    if (any(still_na)) {
      short_mask <- mask_short_na_runs(v, maxgap_kalman)
      apply_mask <- still_na & short_mask
      if (any(apply_mask)) {
        kal <- safe_kalman_struct(v)
        out[apply_mask] <- kal[apply_mask]
      }
    }
  }
```

```
  out
}

to_text_ymd_hms <- function(x) {
  if (inherits(x,"POSIXct") || inherits(x,"POSIXt") || inherits(x,"Date")) {
    return(format(as.POSIXct(x, tz="America/Monterrey"), "%Y-%m-%d
%H:%M:%S"))
  }
  if (is.numeric(x)) {
    return(format(as.POSIXct(x*86400, origin="1899-12-30",
tz="America/Monterrey"), "%Y-%m-%d %H:%M:%S"))
  }
  as.character(x)
}
```

En este bloque se definen funciones de apoyo para el preprocesamiento de series temporales. El objetivo es poder identificar tramos cortos de datos faltantes e imputarlos de manera controlada: primero con interpolación lineal en huecos pequeños y después, en intervalos un poco mayores, con un filtro de Kalman basado en modelos estructurales. Los huecos largos se mantienen como NA para no introducir información artificial. Además, se incluye una rutina para estandarizar la columna de fechas y expresarla en un formato uniforme legible ("YYYY-mm-dd HH:MM:SS"), evitando problemas con números seriales de Excel.

```
input_file <- c("DATOS HISTÓRICOS 2023_2024_TODAS ESTACIONES_ITESM.xlsx")

maxgap_linear  <- 6   # horas
maxgap_kalman  <- 24  # horas
```

Este fragmento define los insumos y parámetros básicos del preprocesamiento. Primero se especifica el archivo original que contiene todas las series de calidad del aire y meteorología. Después se establecen los umbrales de imputación: huecos de hasta 6 horas se rellenan mediante interpolación lineal y vacíos de hasta 24 horas se pueden imputar con el filtro de Kalman. Esto permite un control claro de hasta qué punto se reconstruyen las series sin alterar en exceso su variabilidad real.

```
arreglar_hoja_maestra <- function(input_file,
                                  sheet = "Param_horarios_Estaciones",
                                  output_file = NULL,
                                  maxgap_linear = 6,
                                  maxgap_kalman = 24) {
  if (is.null(output_file)) {
    base <- tools::file_path_sans_ext(basename(input_file))
    output_file <- paste0(base, "_CORREGIDO.xlsx")
  }

  raw <- read_excel(input_file, sheet = sheet, col_names = FALSE,
                    na = c("", "NULL","null","NaN","NA"))
  if (nrow(raw) < 3) stop("La hoja no tiene las 2 filas de encabezado +
```

```
datos.")

  ncols <- ncol(raw)
  vars  <- as.character(unlist(raw[1, 1:ncols]))
  units <- as.character(unlist(raw[2, 1:ncols]))

  if (!grepl("date|fecha|hora", tolower(vars[1] %||% ""))) {
    message("Aviso: la primera columna no luce como fecha; se forzará a
'date'.")
  }
  vars[1]  <- "date"
  if (is.na(units[1]) || units[1] == "") units[1] <- "-"

  if (length(vars)  < ncols) vars  <- c(vars,  paste0("col_", seq_len(ncols -
length(vars))))
  if (length(units) < ncols) units <- c(units, rep("-", ncols -
length(units)))

  dat <- raw[-c(1,2), , drop = FALSE]
  vars_unique <- make.unique(vars, sep = "_")
  names(dat) <- vars_unique

  all_na_col <- vapply(dat, function(x) all(is.na(x)), logical(1))
  empty_name <- vars_unique == "" | is.na(vars_unique)
  keep <- !(all_na_col | empty_name)
  dat          <- dat[,  keep, drop = FALSE]
  vars_unique  <- vars_unique[keep]
  units        <- units[keep]

  if (!"date" %in% vars_unique) stop("No se encontró columna 'date' tras
normalizar encabezados.")

  date_col <- which(vars_unique == "date")[1]
  ord <- order(suppressWarnings(as.numeric(dat[[date_col]])), na.last = TRUE)
  dat <- dat[ord, , drop = FALSE]

  num_cols <- setdiff(names(dat), "date")
  for (col in num_cols) {
    dat[[col]] <- suppressWarnings(as.numeric(dat[[col]]))
  }

  corrected <- dat
  for (col in num_cols) {
    v <- corrected[[col]]
    if (is.numeric(v)) corrected[[col]] <- fill_two_stage(v, maxgap_linear,
maxgap_kalman)
  }
```

```
  corrected$date <- to_text_ymd_hms(corrected$date)

  options("openxlsx.na.string" = NA)
  wb <- createWorkbook()
  addWorksheet(wb, sheet)

  writeData(wb, sheet, t(vars_unique), startRow = 1, startCol = 1,
            colNames = FALSE, rowNames = FALSE)

  writeData(wb, sheet, t(units),       startRow = 2, startCol = 1,
            colNames = FALSE, rowNames = FALSE)

  writeData(wb, sheet, corrected,      startRow = 3, startCol = 1,
            colNames = FALSE, rowNames = FALSE, na.string = NA)

  date_style <- openxlsx::createStyle(numFmt = "yyyy-mm-dd hh:mm:ss")
  openxlsx::addStyle(
    wb, sheet,
    style = date_style,
    rows = 3:(nrow(corrected) + 2),
    cols = 1,
    gridExpand = TRUE
)


  saveWorkbook(wb, output_file, overwrite = TRUE)
  message("Listo: ", output_file)
  return(output_file)
}
```

Este bloque define la función arreglar_hoja_maestra, pensada para limpiar y estandarizar la hoja “maestra” del archivo original que tiene dos filas de encabezado (variables y unidades). El flujo es: primero se leen crudos los encabezados y datos para preservar toda la estructura original; después se normaliza la columna de fechas, se eliminan columnas vacías o duplicadas y se ordena la serie temporal. A continuación se convierten los contaminantes y variables meteorológicas a numéricos y se reconstruyen huecos cortos con el esquema de interpolación lineal y filtro de Kalman. Finalmente, la columna de fecha se reescribe en formato legible y todo se exporta a Excel manteniendo en las dos primeras filas los encabezados y unidades originales. En resumen: esta función prepara la hoja base para que quede consistente, imputada y lista para análisis posteriores sin perder la información de metadatos.

```
# Correr
arreglar_hoja_maestra(input_file,
                      sheet = "Param_horarios_Estaciones",
                       output_file = NULL,
                       maxgap_linear = 6,
                       maxgap_kalman = 24)
```

```
## New names:
## Aviso: la primera columna no luce como fecha; se forzará a 'date'.
## • `` -> `...1`
## • `` -> `...2`
## • `` -> `...3`
## • `` -> `...4`
## • `` -> `...5`
## • `` -> `...6`
## • `` -> `...7`
## • `` -> `...8`
## • `` -> `...9`
## • `` -> `...10`
## • `` -> `...11`
## • `` -> `...12`
## • `` -> `...13`
## • `` -> `...14`
## • `` -> `...15`
## • `` -> `...16`
## • `` -> `...17`
## • `` -> `...18`
## • `` -> `...19`
## • `` -> `...20`
## • `` -> `...21`
## • `` -> `...22`
## • `` -> `...23`
## • `` -> `...24`
## • `` -> `...25`
## • `` -> `...26`
## • `` -> `...27`
## • `` -> `...28`
## • `` -> `...29`
## • `` -> `...30`
## • `` -> `...31`
## • `` -> `...32`
## • `` -> `...33`
## • `` -> `...34`
## • `` -> `...35`
## • `` -> `...36`
## • `` -> `...37`
## • `` -> `...38`
## • `` -> `...39`
## • `` -> `...40`
## • `` -> `...41`
## • `` -> `...42`
## • `` -> `...43`
## • `` -> `...44`
## • `` -> `...45`
## • `` -> `...46`
## • `` -> `...47`
## • `` -> `...48`
```

```
## • `` -> `...49`
## • `` -> `...50`
## • `` -> `...51`
## • `` -> `...52`
## • `` -> `...53`
## • `` -> `...54`
## • `` -> `...55`
## • `` -> `...56`
## • `` -> `...57`
## • `` -> `...58`
## • `` -> `...59`
## • `` -> `...60`
## • `` -> `...61`
## • `` -> `...62`
## • `` -> `...63`
## • `` -> `...64`
## • `` -> `...65`
## • `` -> `...66`
## • `` -> `...67`
## • `` -> `...68`
## • `` -> `...69`
## • `` -> `...70`
## • `` -> `...71`
## • `` -> `...72`
## • `` -> `...73`
## • `` -> `...74`
## • `` -> `...75`
## • `` -> `...76`
## • `` -> `...77`
## • `` -> `...78`
## • `` -> `...79`
## • `` -> `...80`
## • `` -> `...81`
## • `` -> `...82`
## • `` -> `...83`
## • `` -> `...84`
## • `` -> `...85`
## • `` -> `...86`
## • `` -> `...87`
## • `` -> `...88`
## • `` -> `...89`
## • `` -> `...90`
## • `` -> `...91`
## • `` -> `...92`
## • `` -> `...93`
## • `` -> `...94`
## • `` -> `...95`
## • `` -> `...96`
## • `` -> `...97`
## • `` -> `...98`
```

```
## • `` -> `...99`
## • `` -> `...100`
## • `` -> `...101`
## • `` -> `...102`
## • `` -> `...103`
## • `` -> `...104`
## • `` -> `...105`
## • `` -> `...106`
## • `` -> `...107`
## • `` -> `...108`
## • `` -> `...109`
## • `` -> `...110`
## • `` -> `...111`
## • `` -> `...112`
## • `` -> `...113`
## • `` -> `...114`
## • `` -> `...115`
## • `` -> `...116`
## • `` -> `...117`
## • `` -> `...118`
## • `` -> `...119`
## • `` -> `...120`
## • `` -> `...121`
## • `` -> `...122`
## • `` -> `...123`
## • `` -> `...124`
## • `` -> `...125`
## • `` -> `...126`
## • `` -> `...127`
## • `` -> `...128`
## • `` -> `...129`
## • `` -> `...130`
## • `` -> `...131`
## • `` -> `...132`
## • `` -> `...133`
## • `` -> `...134`
## • `` -> `...135`
## • `` -> `...136`
## • `` -> `...137`
## • `` -> `...138`
## • `` -> `...139`
## • `` -> `...140`
## • `` -> `...141`
## • `` -> `...142`
## • `` -> `...143`
## • `` -> `...144`
## • `` -> `...145`
## • `` -> `...146`
## • `` -> `...147`
## • `` -> `...148`
```

```
## • `` -> `...149`
## • `` -> `...150`
## • `` -> `...151`
## • `` -> `...152`
## • `` -> `...153`
## • `` -> `...154`
## • `` -> `...155`
## • `` -> `...156`
## • `` -> `...157`
## • `` -> `...158`
## • `` -> `...159`
## • `` -> `...160`
## • `` -> `...161`
## • `` -> `...162`
## • `` -> `...163`
## • `` -> `...164`
## • `` -> `...165`
## • `` -> `...166`
## • `` -> `...167`
## • `` -> `...168`
## • `` -> `...169`
## • `` -> `...170`
## • `` -> `...171`
## • `` -> `...172`
## • `` -> `...173`
## • `` -> `...174`
## • `` -> `...175`
## • `` -> `...176`
## • `` -> `...177`
## • `` -> `...178`
## • `` -> `...179`
## • `` -> `...180`
## • `` -> `...181`
## • `` -> `...182`
## • `` -> `...183`
## • `` -> `...184`
## • `` -> `...185`
## • `` -> `...186`
## • `` -> `...187`
## • `` -> `...188`
## • `` -> `...189`
## • `` -> `...190`
## • `` -> `...191`
## • `` -> `...192`
## • `` -> `...193`
## • `` -> `...194`
## • `` -> `...195`
## • `` -> `...196`
## • `` -> `...197`
## • `` -> `...198`
```

```
## • `` -> `...199`
## • `` -> `...200`
## • `` -> `...201`
## • `` -> `...202`
## • `` -> `...203`
## • `` -> `...204`
## • `` -> `...205`
## • `` -> `...206`
## • `` -> `...207`
## • `` -> `...208`
## • `` -> `...209`
## • `` -> `...210`
## • `` -> `...211`
## • `` -> `...212`
## • `` -> `...213`
## • `` -> `...214`
## • `` -> `...215`
## • `` -> `...216`
## • `` -> `...217`
## • `` -> `...218`
## • `` -> `...219`
## • `` -> `...220`
## • `` -> `...221`
## • `` -> `...222`
## • `` -> `...223`
## • `` -> `...224`
## • `` -> `...225`
## • `` -> `...226`
## • `` -> `...227`
## • `` -> `...228`
## • `` -> `...229`
## • `` -> `...230`
## • `` -> `...231`
## • `` -> `...232`
## • `` -> `...233`
## • `` -> `...234`
## • `` -> `...235`
## • `` -> `...236`
## • `` -> `...237`
## • `` -> `...238`
## • `` -> `...239`
## • `` -> `...240`

## Warning: The `value` argument of `names<-()` can't be empty as of tibble
3.0.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Listo: DATOS HISTÓRICOS 2023_2024_TODAS ESTACIONES_ITESM_CORREGIDO.xlsx
```

```
## [1] "DATOS HISTÓRICOS 2023_2024_TODAS ESTACIONES_ITESM_CORREGIDO.xlsx"

# Ver un log de rellenos (opcional)

input_file  <- "DATOS HISTÓRICOS 2023_2024_TODAS
ESTACIONES_ITESM_CORREGIDO.xlsx"
output_file <- sub("\\.xlsx$", "_POSTQC.xlsx", input_file)

PM10_CAP <- 1000
O3_CAP   <- 400
NA_DROP_FRAC <- 0.70

sheets <- excel_sheets(input_file)
wb <- createWorkbook()

for (sh in sheets) {
  message("Procesando hoja: ", sh)
  df <- read_excel(input_file, sheet = sh)

  if (!is.data.frame(df) || ncol(df) <= 1) {
    addWorksheet(wb, sh); writeData(wb, sh, df); next
  }

  if ("date" %in% names(df)) {
    d <- df[["date"]]

    d_chr <- as.character(d)
    d_chr[trimws(d_chr) %in% c("-", "", "NA", "NaN", "null", "NULL")] <- NA

    has_comma <- grepl(",", d_chr, fixed = TRUE)
    d_chr[has_comma] <- gsub(",", ".", d_chr[has_comma], fixed = TRUE)

    dn <- suppressWarnings(as.numeric(d_chr))
    frac_num <- mean(!is.na(dn))
    if (frac_num > 0.6 || all(!is.na(dn))) {
      d_posix <- as.POSIXct(dn * 86400, origin = "1899-12-30", tz =
"America/Mexico_City")
    } else {
      d_posix <- suppressWarnings(parse_date_time(
        d_chr,
        orders = c(
          "Y-m-d H:M:S", "Y-m-d H:M",
          "Y/m/d H:M:S", "Y/m/d H:M",
          "d/m/Y H:M:S", "d/m/Y H:M",
          "m/d/Y H:M:S", "m/d/Y H:M",
          "Y-m-d", "Y/m/d", "d/m/Y", "m/d/Y"
        ),
        tz = "America/Mexico_City",
        exact = FALSE
```

```
      ))
      if (mean(!is.na(d_posix)) < 0.5 && any(!is.na(dn))) {
        d_posix <- as.POSIXct(dn * 86400, origin = "1899-12-30", tz =
"America/Mexico_City")
      }
    }

    df[["date"]] <- d_posix
  }

  if ("PM10" %in% names(df)) {
    num <- suppressWarnings(as.numeric(df[["PM10"]]))
    idx <- !is.na(num) & num > PM10_CAP
    if (any(idx)) df[idx, "PM10"] <- NA
  } else if ("pm10" %in% names(df)) {
    num <- suppressWarnings(as.numeric(df[["pm10"]]))
    idx <- !is.na(num) & num > PM10_CAP
    if (any(idx)) df[idx, "pm10"] <- NA
  }

  if ("O3" %in% names(df)) {
    num <- suppressWarnings(as.numeric(df[["O3"]]))
    idx <- !is.na(num) & num > O3_CAP
    if (any(idx)) df[idx, "O3"] <- NA
  } else if ("o3" %in% names(df)) {
    num <- suppressWarnings(as.numeric(df[["o3"]]))
    idx <- !is.na(num) & num > O3_CAP
    if (any(idx)) df[idx, "o3"] <- NA
  }

  na_frac <- sapply(df, function(x) mean(is.na(x)))
  to_drop <- setdiff(names(df)[na_frac > NA_DROP_FRAC], "date")
  if (length(to_drop)) {
    df <- df[, setdiff(names(df), to_drop), drop = FALSE]
  }

  addWorksheet(wb, sh)
  writeData(wb, sh, df)

  if ("date" %in% names(df) && inherits(df[["date"]], "POSIXct")) {
    date_col <- which(names(df) == "date")
    if (length(date_col) == 1) {
      date_style <- createStyle(numFmt = "yyyy-mm-dd hh:mm:ss")
      addStyle(wb, sh, style = date_style,
               rows = 2:(nrow(df) + 1), cols = date_col,
               gridExpand = TRUE, stack = TRUE)
    }
  }
}
```

```
## Procesando hoja: Param_horarios_Estaciones

saveWorkbook(wb, output_file, overwrite = TRUE)
message("Listo ✓ Guardado como: ", output_file)

## Listo ✓ Guardado como: DATOS HISTÓRICOS 2023_2024_TODAS
ESTACIONES_ITESM_CORREGIDO_POSTQC.xlsx
```

Este bloque recorre todas las hojas del archivo ya corregido para aplicar un control de calidad final y estandarizar la columna temporal antes de guardar un libro nuevo. Primero, detecta y normaliza la variable date: limpia símbolos o cadenas ambiguas, convierte seriales de Excel a fecha-hora local y, si vienen como texto, intenta varios formatos comunes hasta obtener un POSIXct consistente; además, aplica un formato de visualización en Excel para que la fecha se vea como “yyyy-mm-dd hh:mm:ss”. Después, implementa reglas mínimas de depuración sobre outliers instrumentales: valores de PM10 por encima de 1000 µg/m^3 y de O3 por encima de 400 ppb se recodifican como faltantes para evitar que picos no plausibles sesguen los análisis. Finalmente, elimina columnas con más de 70% de datos faltantes (conservando siempre la columna date), y escribe cada hoja depurada en un nuevo archivo, manteniendo nombres y estructura originales. Con esto se asegura una base homogénea, sin extremos espurios ni variables prácticamente vacías.