

```

# =====
#
# SCRIPT: ARGOS_CLEANING
#
# AUTHOR: Raphaël Musseau – BioSphère Environnement
# YEAR: 2025
# LICENSE: MIT
#
# DESCRIPTION:
# Lit des données ARGOS et conserve une seule localisation par "passage" opérationnel.
#
# Passage satellite (définition opérationnelle) :
# - pour un même individu (N° ID) et un même satellite (Sat.),
# un passage est défini comme un ensemble de localisations partageant le même
# Loc idx et séparées par un intervalle temporel ≤ gap_min (15 minutes).
# Le même Loc idx pouvant être recyclé par ARGOS, y compris au cours d'une
# même journée et pour un même satellite, les localisations sont scindées
# en passages distincts dès que l'intervalle temporel dépasse ce seuil.
#
# Sélection de la localisation conservée (1 par passage) :
# 1) meilleure classe ARGOS (3 > 2 > 1 > 0 > A > B > C)
# 2) plus petite ellipse (demi-grand axe)
# 3) message le plus récent (Date Msg., sinon Date de loc.)
#
# SORTIE PRINCIPALE : d_final (data.frame) + fichier .csv sauvegardé (argos_clean.csv)
#
# =====

## PACKAGES ----
suppressPackageStartupMessages({
  library(dplyr)
  library(lubridate)
  library(tcltk)
})

## 1) PARAMÈTRES & SOURCE DES DONNÉES ----
csv_path <- "H:/Dossier_x/TIMILLA_Jeu_donnees_test_ARGOS.csv"

# Nouveau "épisode" si dt > gap_min (minutes), même individu + même satellite + même Loc idx
gap_min <- 15

# Ordre de qualité ARGOS (meilleur -> moins bon)
lc_order <- c("3","2","1","0","A","B","C")
lc_rank <- setNames(seq_along(lc_order), lc_order)

## 2) FONCTIONS UTILITAIRES ----
to_num <- function(x) suppressWarnings(as.numeric(gsub(",", ".", as.character(x))))

to_time <- function(x) {
  x <- trimws(as.character(x))
  x[x == ""] <- NA_character_
  x <- sub(".*$", "", x)
  x <- gsub("T", " ", x, fixed = TRUE)

  suppressWarnings(as.POSIXct(
    x, tz = "UTC",
    tryFormats = c(
      "%Y-%m-%d %H:%M:%S",
      "%Y-%m-%d %H:%M",
      "%d/%m/%Y %H:%M:%S",
      "%d/%m/%Y %H:%M"
    )
  ))
}

```

```

}

## 3) LECTURE DU CSV SOURCE ----
raw <- read.csv(
  csv_path,
  sep = ";",
  stringsAsFactors = FALSE,
  check.names = FALSE,
  fileEncoding = "UTF-8"
)

required <- c("N° ID", "Date de loc.", "Longitude", "Latitude", "Qualité loc.", "Sat.", "Loc idx")
if (!all(required %in% names(raw))) {
  stop("Colonnes requises manquantes (il faut notamment 'Loc idx'). Trouvées :\n",
       paste(names(raw), collapse = " | "))
}

has_smaj <- "Demi-grand axe" %in% names(raw)
has_date_msg <- "Date Msg" %in% names(raw)

## 4) PRÉPARATION & NORMALISATION ----
d <- tibble(
  id = as.character(raw[["N° ID"]]),
  `Sat.` = toupper(trimws(as.character(raw[["Sat."]]))),
  loc_idx = suppressWarnings(as.integer(to_num(raw[["Loc idx"]]))),
  date_loc = to_time(raw[["Date de loc."]]),
  lon = to_num(raw[["Longitude"]]),
  lat = to_num(raw[["Latitude"]]),
  lc = toupper(trimws(as.character(raw[["Qualité loc."]])))
)

# Champs optionnels (utiles uniquement pour départager)
if (has_smaj) {
  smaj_raw <- to_num(raw[["Demi-grand axe"]])
  d$smaj_km <- if (isTRUE(stats::median(smaj_raw, na.rm = TRUE) > 50)) smaj_raw/1000 else smaj_raw
} else {
  d$smaj_km <- NA_real_
}

if (has_date_msg) {
  d$date_msg <- to_time(raw[["Date Msg"]])
} else {
  d$date_msg <- as.POSIXct(NA, tz = "UTC")
}

# Rang LC (LC inconnue -> classée après)
d <- d %>%
  mutate(
    lc_rank = lc_rank[lc],
    lc_rank = ifelse(is.na(lc_rank), length(lc_order) + 1L, lc_rank)
  )

## 5) PRÉ-FILTRAGE ----
d <- d %>%
  filter(!is.na(id), !is.na(date_loc), is.finite(lon), is.finite(lat)) %>%
  filter(!is.na(`Sat.`), `Sat.` != "") %>%
  filter(!is.na(loc_idx)) %>% # sans Loc idx => pas de "passage" définissable ici
  arrange(id, `Sat.`, loc_idx, date_loc)

## 6) DÉFINITION DES PASSAGES (Loc idx + continuité temporelle) ----
# Même id + même Sat. + même loc_idx :
# - on découpe en épisodes si dt > gap_min (loc_idx recyclé dans la journée / etc.)
d_pass <- d %>%

```

```

group_by(id, `Sat.` , loc_idx) %>%
arrange(date_loc, .by_group = TRUE) %>%
mutate(
  dt_min = as.numeric(difftime(date_loc, lag(date_loc), units = "mins")),
  new_ep = ifelse(is.na(dt_min) | dt_min > gap_min, 1L, 0L),
  ep_id = cumsum(new_ep)
) %>%
ungroup()

## 7) UNE UNIQUE LOCALISATION CONSERVÉE PAR PASSAGE (id+sat+loc_idx+ep) ----
d_final <- d_pass %>%
group_by(id, `Sat.` , loc_idx, ep_id) %>%
arrange(
  lc_rank,
  is.na(smaj_km), smaj_km,
  dplyr::desc(coalesce(date_msg, date_loc)),
  .by_group = TRUE
) %>%
slice(1) %>%
ungroup() %>%
select(-lc_rank, -dt_min, -new_ep, -ep_id)

## 8) RÉSUMÉ CONSOLE ----
cat("\n=== ARGOS CLEANING – RÉSUMÉ ===\n")
cat("Lignes brutes      :", nrow(raw), "\n")
cat("Après pré-filtrage minimal      :", nrow(d), "\n")
cat("Final (1 loc / passage loc_idx+ep) :", nrow(d_final), "\n\n")

cat("Effectifs par LC (final) :\n")
print(table(d_final$lc, useNA = "ifany"))

cat("\nAperçu d_final :\n")
print(utils::head(d_final, 10))

## 9) SAUVEGARDE AVEC FENÊTRE (argos_clean.csv) ----
cat("\nUne fenêtre va s'ouvrir pour choisir où enregistrer le fichier .csv...\n")

outfile_tcl <- tcltk::tkgetSaveFile(
  filetypes = "{{CSV Files} {.csv}} {{All files} *}",
  defaultextension = ".csv",
  initialfile = "argos_clean.csv"
)

outfile <- as.character(outfile_tcl)

if (length(outfile) == 1 && nzchar(outfile)) {
  write.csv(d_final, outfile, row.names = FALSE, fileEncoding = "UTF-8")
  cat("\nFichier enregistré dans :\n", outfile, "\n")
} else {
  cat("\nSauvegarde annulée par l'utilisateur.\n")
}

## FIN DU SCRIPT

```