Assignment 2 - Reading On-line Data and Visualizing Hurricane Tracks

EVR-5086 Fall 2025

Adyan Rios

2025-09-16

Table of contents

Assignment 2 - Reading On-line Data and Visualizing Hurricane Tracks

1 Data Retrieval and Parsing

The purpose of this code is to access and format the HURDAT2 dataset into an analysis-ready format. Since I expect to explore these data further, I did not want to perform wrangling and formatting only on filtered subsets. Instead, I extracted the header information for each storm record, expanded it, and attached it to each corresponding data line.

There are many possible approaches to this task, including base R methods. I chose to use tidyr and dplyr because their piping syntax allows me to write modular code without overwriting objects or cluttering the global environment. While I could have created a single long script, I prefer a modular workflow where each section has a clear intention. For example, this part of the workflow focuses only on data retrieval and formatting.

At the end of Section 1, I save the resulting data frame as an .RDS file. I prefer RDS over RData because RDS requires explicit object naming when loaded, which supports better coding practices and clearer, more reproducible code.

The steps of the code are annotated in the code chunks, and the first chunk defines and loads all the required libraries used in Section 1. One limitation of my approach is that effective use or contribution requires familiarity with GitHub, RStudio, and Quarto.

```
# Check if libraries are installed; install if not.
if (!require("pacman")) install.packages("pacman")
pacman::p_load(here, curl, tidyr, dplyr, lubridate)
```

```
# Specify the source and output file name and location
url <- "https://www.nhc.noaa.gov/data/hurdat/hurdat2-1851-2024-040425.txt"
destfile <- here("assignment2","hurdat.txt")

# Download and save the dataset
curl_download(url = url, destfile = destfile)
```

```
# Read in data and differentiate between headers and data lines
lines <- readLines("hurdat.txt")                      # Read text file

# Prep headers
storm_headers <- lines[grepl("^AL", lines)]           # Keep only storm headers
header_parts <- strsplit(storm_headers, ",")          # Split based on ","
header_matrix <- do.call(rbind, header_parts)         # Convert to matrix
header_df <- as.data.frame(header_matrix)             # Convert to data frame
colnames(header_df) <- c("storm_id", "name", "rows") # Name columns

# Repeat each header based on the 'rows' column in tidyr
header_expand <- header_df |>
  mutate(rows = as.numeric(rows)) |>
  uncount(rows)

# Prep data
storm_data <- lines[!grepl("^AL", lines)] # Keep only data
hurdat_parts <- strsplit(storm_data, ",")   # Split based on ","
hurdat_matrix <- do.call(rbind, hurdat_parts) # Convert to matrix
hurdat_df <- as.data.frame(hurdat_matrix)     # Convert to to data frame

hurdat_fields <- c("yyyymmdd", "hhmm", "record", "status",
                   "lat_hemi", "lon_hemi", "wind", "pressure",
                   "ne34", "se34", "sw34", "nw34",
```

```
                  "ne50", "se50", "sw50", "nw50",
                  "ne64", "se64", "sw64", "nw64", "radius")

colnames(hurdat_df) <- hurdat_fields # Name columns

# Build analysis ready data set
hurdat_ar <- header_expand |>
  bind_cols(hurdat_df) |> # Glue together storm id and name with data
  mutate(
    yyyymmdd = ymd(yyyymmdd),                    # Tell R this is a date
    hhmm = strptime(hhmm, format = "%H%M"),      # Tell R this is a time
    hhmm = format(hhmm, "%H:%M"),
    lat = as.numeric(substr(lat_hemi, 1, nchar(lat_hemi)-1)), # Remove "S"
    lon = as.numeric(substr(lon_hemi, 1, nchar(lon_hemi)-1)), # Remove "W"
    lat_hemi = substr(lat_hemi, nchar(lat_hemi), nchar(lat_hemi)),
    lon_hemi = substr(lon_hemi, nchar(lon_hemi), nchar(lon_hemi)),
    lat = if_else(lat_hemi == "S", -lat, lat), # Make lat negative if "S"
    lon = if_else(lon_hemi == "W", -lon, lon)  # Make lon negative if "W"
  ) |>
  mutate_at(c(9:25), as.numeric) |>                  # Make data numeric
  mutate(across(where(is.numeric), ~na_if(., -999))) # Replace NAs

# Save formatted data to read into next quarto environment
saveRDS(hurdat_ar, file = here("assignment2", "hurdat.rds"))
```

2 Data Visualization

The data visualization has several exciting features. Because the data set was already formatted into an analysis-ready structure, this part of the assignment focuses only on visualizing a given storm ID. It begins with defining and loading the packages used later in the code. The leaflet and webshot2 packages were new to me. Leaflet allows me to create an interactive map, similar to folium in Python.

Since I am rendering my report to both HTML and PDF, I needed different approaches for each format. In HTML, I was able to embed the leaflet map directly as an htmlwidget. For PDF output, I learned to use conditional content so the widget only displays in HTML, and a static snapshot (generated with webshot2) is included in the PDF. For both versions, I provided a figure caption and alt text to improve clarity and accessibility.

To add more complexity and depth to the track visualization, I incorporated a color scale representing wind speed. This makes the map more informative and highlights storm intensity changes along its path. I think interactive widgets can serve as a useful precursor to fully

developed applications for data visualization. I am excited about the continued advancements in interactive figures which allow non-coders to explore and interact with data in more meaningful ways. Although I did not implement dynamic selection in the HTML rendering of this assignment, I looked into some of the latest developments in Quarto Dashboards. For now, I set up an optional user input similar to what we did in Python. When the R code is run interactively, the user is prompted to provide a storm name; otherwise, a default storm ID is used to ensure the code still runs smoothly during rendering.

```
# Check if libraries are installed; install if not.
if (!require("pacman")) install.packages("pacman")
pacman::p_load(here, stringr, leaflet, webshot2, dplyr)
```

```
default_name = "AL092021"

# If interactive (R console / RStudio), ask the user
if (interactive()) {
  storm_id <- readline(
    prompt = paste0("Enter a storm ID using ALnnyyy format [default = ",   default_name, "]:
    )
  if (storm_id == "") storm_id <- default_name
} else {
  # If running non-interactively (e.g., knitting to PDF/HTML), use default
  storm_id <- default_name
}
```

```
# Read in data
hurdat_ar <- readRDS(here("assignment2", "hurdat.Rds"))

# Create a reusable function
track_storm <- function(dat, storm_id, zoom = 4,
                            init_location = c(20, -50)) {
  # Filter and order the points for the selected storm
  storm <- dat |>
    filter(storm_id == !!storm_id, !is.na(lat), !is.na(lon)) |>
    mutate(status = str_trim(status)) |>
    arrange(yyyymmdd, hhmm)

  if (nrow(storm) == 0) stop("No points found for this storm_id.")

  # Build popup: date + time + status (e.g., "1851-06-25 00:00 - HU")
  popup_txt <- paste0(
    format(storm$yyyymmdd, "%Y-%m-%d"), " ", storm$hhmm,
```

```
    " - ", storm$status
  )

  # Definecolor range
  pal <- colorNumeric(
    palette = "YlOrRd",   # yelloe = weak winds, red = strong winds
    domain = storm$wind   # The range of wind speeds
  )

  # Create map
  m <- leaflet(storm) |>
  addTiles() |>
  addPolylines(lng = ~lon, lat = ~lat, color = "blue",
               weight = 2.5, opacity = 1) |>
  addCircleMarkers(
    lng = ~lon,
    lat = ~lat,
    color = ~pal(wind),       # marker color by wind
    radius = 5,               # size of marker
    stroke = FALSE,
    fillOpacity = 0.8,
    popup = ~paste0(format(yyyymmdd, "%Y-%m-%d"), " ", hhmm,
                    "<br>Wind: ", wind, " kt",
                    "<br>Status: ", status)
  ) |>
  addLegend(
    "bottomright",
    pal = pal,
    values = ~wind,
    title = "Wind (kt)",
    opacity = 1
  )

  file_html <- here("assignment2", paste0(storm_id, "_map.html"))
  htmlwidgets::saveWidget(m, file_html, selfcontained = TRUE)
  m
}

leaflet_png <- function(m) {
  file_png = here("assignment2", "hurricane_tracks_map.png")
  html_tmp <- here("assignment2", "hurricane_tracks_map_tmp.html")
  htmlwidgets::saveWidget(m, html_tmp, selfcontained = TRUE)
```

```
  webshot2::webshot(html_tmp, file = file_png, vwidth = 1400,
                    vheight = 900, zoom = 1)
  return(file_png)
}
```

```
m <- track_storm(hurdat_ar, storm_id = storm_id)
png_file <- leaflet_png(m)
knitr::include_graphics(png_file)
```

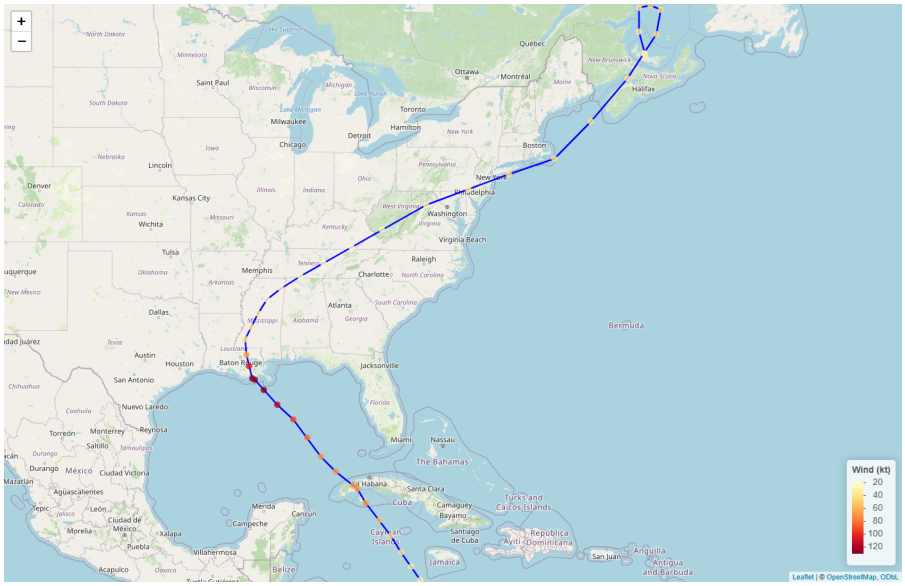

Figure 1: Static storm track AL092021 (PDF).

3 Links to Colab and GitHub

Assignment 2 Google Colab

Quarto book chapter on GitHub