# =========================================================

# 🏇 Horse Racing Scraper + Historical Merger (QLD Focus)

# =========================================================

# --- 0. Load Required Packages ---

packages <- c("rvest", "dplyr", "stringr", "tcltk", "httr", "xml2", "readr")

new\_packages <- packages[!(packages %in% installed.packages()[, "Package"])]

if (length(new\_packages)) install.packages(new\_packages, dependencies = TRUE)

lapply(packages, library, character.only = TRUE)

# --- 1. Helpers ---

safe\_read\_html <- function(url) {

tryCatch(read\_html(GET(url, config = config(ssl\_verifypeer = 0L))), error = function(e) NULL)

}

normalize\_name <- function(x) {

x %>%

toupper() %>%

str\_trim() %>%

str\_replace\_all("[^A-Z0-9 ]", "") %>%

str\_squish()

}

load\_horse\_history <- function() {

file\_path <- tk\_choose.files(caption = "Select Historical File (Combined\_\*.csv)")

if (length(file\_path) == 0) return(NULL)

read\_csv(file\_path, show\_col\_types = FALSE) %>%

mutate(HorseNameClean = normalize\_name(HorseName))

}

# --- 2. UI: State + Venue + Date ---

select\_state <- function() {

cal\_url <- "https://www.racingaustralia.horse/FreeFields/Calendar.aspx"

page <- safe\_read\_html(cal\_url)

state\_codes <- page %>%

html\_nodes("a") %>%

html\_attr("href") %>%

.[grepl("/FreeFields/Calendar.aspx\\?State=", .)] %>%

str\_extract("(?<=State=)[A-Z]+") %>%

unique()

selected\_state <- tclVar(state\_codes[1])

tt <- tktoplevel()

tkwm.title(tt, "Select a State")

tkgrid(tklabel(tt, text = "Choose a state:"), padx = 10, pady = 10)

dropdown <- ttkcombobox(tt, values = state\_codes, textvariable = selected\_state)

tkgrid(dropdown, padx = 10, pady = 10)

tkgrid(tkbutton(tt, text = "OK", command = function() tkdestroy(tt)), padx = 10, pady = 10)

tkwait.window(tt)

tclvalue(selected\_state)

}

select\_venue <- function(state) {

calendar\_url <- paste0("https://www.racingaustralia.horse/FreeFields/Calendar.aspx?State=", state)

calendar\_page <- safe\_read\_html(calendar\_url)

race\_table <- calendar\_page %>% html\_node("table.race-fields") %>% html\_table(fill = TRUE)

df <- as.data.frame(race\_table, stringsAsFactors = FALSE)

venue\_dates <- df[[1]]

venue\_names <- df[[2]]

popup <- tktoplevel()

tkwm.title(popup, paste("Venues for", state))

tkgrid(tklabel(popup, text = "Double-click a venue to view races:"), padx = 10, pady = 10)

listbox <- tklistbox(popup, height = 20, width = 50, selectmode = "single")

for (i in seq\_along(venue\_names)) {

tkinsert(listbox, "end", paste(venue\_dates[i], "-", venue\_names[i]))

}

tkgrid(listbox)

selection <- new.env(parent = emptyenv())

selection$made <- FALSE

tkbind(listbox, "<Double-Button-1>", function() {

idx <- as.integer(tkcurselection(listbox))

selection$venue <- venue\_names[idx + 1]

selection$date\_str <- str\_extract(venue\_dates[idx + 1], "\\d{1,2}-[A-Za-z]{3}")

selection$made <- TRUE

tkdestroy(popup)

})

tkwait.window(popup)

if (isTRUE(selection$made)) selection else NULL

}

# --- 3. Scraper: Full Race Card (HTML-anchored)

scrape\_full\_meeting <- function(state, venue, date\_str) {

date\_obj <- as.Date(paste0(date\_str, "-", format(Sys.Date(), "%Y")), format = "%d-%b-%Y")

formatted\_date <- format(date\_obj, "%Y%b%d")

nice\_date <- format(date\_obj, "%Y-%m-%d")

url\_venue <- URLencode(venue, reserved = TRUE)

meeting\_url <- paste0("https://www.racingaustralia.horse/FreeFields/Form.aspx?Key=",

formatted\_date, ",", state, ",", url\_venue)

page <- safe\_read\_html(meeting\_url)

if (is.null(page)) return(NULL)

race\_blocks <- html\_nodes(page, xpath = "//table[contains(@class,'race-strip-fields')]")

race\_headers <- html\_nodes(page, xpath = "//span[@class='raceNum']/parent::\*")

race\_texts <- html\_text(race\_headers, trim = TRUE)

race\_info <- str\_match(race\_texts, "(\\d{1,2}:\\d{2}\\s\*(?:AM|PM)?)\\s+(.\*?)\\s\*\\((\\d{3,4})\\s\*METRES?\\)")

race\_list <- list()

for (i in seq\_along(race\_blocks)) {

tbl <- html\_table(race\_blocks[[i]], fill = TRUE)

df <- as.data.frame(tbl, stringsAsFactors = FALSE)

# Harmonize runner number column to character

if ("No" %in% names(df)) {

df$No <- as.character(df$No)

}

df$Race <- i

df$Time <- race\_info[i, 2]

df$RaceName <- race\_info[i, 3]

df$Distance <- race\_info[i, 4]

race\_list[[i]] <- df

}

race\_data <- bind\_rows(race\_list) %>%

rename(Horse = Horse, Jockey = Jockey, Trainer = Trainer,

Barrier = Barrier, Weight = Weight, Number = No) %>%

mutate(Number = as.integer(Number))

race\_data <- race\_data %>%

mutate(

Date = nice\_date,

Venue = venue,

DistanceNumeric = as.numeric(Distance),

WeightNumeric = as.numeric(str\_extract(Weight, "\\d+\\.?\\d\*")),

HorseNameClean = normalize\_name(Horse)

) %>%

select(Date, Venue, Race, Time, RaceName, DistanceNumeric,

Number, Horse, HorseNameClean, Jockey, Trainer, Barrier, Weight, WeightNumeric)

}

# --- 4. Merge + Save ---

merge\_with\_history <- function(scraped\_df, history\_df) {

left\_join(scraped\_df, history\_df, by = "HorseNameClean")

}

save\_merged\_races <- function(merged\_df, venue) {

save\_dir <- tk\_choose.dir(caption = "Select Folder to Save Merged Race Data")

if (is.na(save\_dir) || save\_dir == "") return()

races <- unique(merged\_df$Race)

for (r in races) {

race\_subset <- merged\_df %>% filter(Race == r)

out\_file <- file.path(save\_dir, paste0("Merged\_", gsub(" ", "\_", venue), "\_R", r, ".csv"))

write\_csv(race\_subset, out\_file)

}

# Save combined file

combined\_file <- file.path(save\_dir, paste0("Merged\_", gsub(" ", "\_", venue), "\_All.csv"))

write\_csv(merged\_df, combined\_file)

tkmessageBox(

title = "Saved",

message = paste0(

"✅ Merged race data saved to:\n", save\_dir, "\n\n",

"📁 Individual races: ", length(races), "\n",

"📦 Combined file: Merged\_", gsub(" ", "\_", venue), "\_All.csv"

)

)

}

# --- 5. Run ---

chosen\_state <- select\_state()

venue\_info <- select\_venue(chosen\_state)

history\_df <- load\_horse\_history()

if (!is.null(venue\_info) && !is.null(history\_df)) {

scraped\_df <- scrape\_full\_meeting(chosen\_state, venue\_info$venue, venue\_info$date\_str)

if (!is.null(scraped\_df)) {

merged\_df <- merge\_with\_history(scraped\_df, history\_df)

save\_merged\_races(merged\_df, venue\_info$venue)

} else {

tkmessageBox(title = "Error", message = "Could not scrape race card.")

}

}

**Overview**

This report documents the execution of a modular R-based pipeline designed to scrape race-day horse data from Racing Australia, merge it with historical performance records, and output structured CSV files for downstream analysis.

Workflow Summary

**Modules Used:**

* **State & Venue Selection:** Interactive UI for selecting race location and date
* **Race Card Scraper:** HTML-anchored extraction of runner-level data across all races
* **Historical Merger:** Cleaned horse names matched against the historical dataset
* **Output Generator:** One CSV per race + one combined CSV for full meeting(longform)

Meeting Details

| **Field** | **Value** |
| --- | --- |
| **State** | QLD |
| **Venue** | *[Selected Venue]* |
| **Race Date** | *[Selected Date]* |
| **Total Races Detected** | *[e.g. 8]* |
| **Historical File Used** | Combined\_\*.csv |

Output Files

**Saved to:** *[Selected Folder]*

| **File Name** | **Description** |
| --- | --- |
| Merged\_[Venue]\_R1.csv | Race 1 runners + historical data |
| Merged\_[Venue]\_R2.csv | Race 2 runners + historical data |
| ... | ... |
| Merged\_[Venue]\_R8.csv | Race 8 runners + historical data |
| **Merged\_[Venue]\_All.csv** | Combined data from all races |

* All files are formatted for downstream modelling, stakeholder review, or dashboard integration.

**Data Fields Included**

Each CSV contains the following columns:

* Date, Venue, Race, Time, RaceName, DistanceNumeric
* Number, Horse, HorseNameClean, Jockey, Trainer, Barrier, Weight, WeightNumeric
* **Historical fields** (e.g. previous wins, earnings, etc. from Combined\_\*.csv)

🧪 Validation Notes

* Race segmentation now uses HTML structure (race-strip-fields) rather than runner number resets
* All races are reliably detected and saved
* Horse names are normalised for accurate historical matching
* Combined file includes all enriched runners across the meeting

📈 Next Steps

* Add odds and scratch status for predictive modelling
* Integrate summary stats (e.g. average weight, barrier bias)
* Build dashboard-ready exports or Shiny interface

From the sample xml file, where do I get the

<https://www.racingaustralia.horse/FreeFields/AllForm.aspx?Key=2025Aug21%2CQLD%2CRockhampton&recentForm>