DATA534: Project Notebook

Mohammad Zaed Iqbal Khan

2026-01-26

21-Jan-2026

Summary

Today I focused on building a prototype pipeline for integrating macro-level contextual data (weather, economic indicators, and news signals) into a unified time-series format that can later be attached to revenue or performance datasets.

I experimented with three core components:

1) Weather Data Integration (Open-Meteo API):

I retrieved hourly temperature and precipitation data for a specified location (Canada) over a defined date range. I then performed data wrangling to aggregate hourly observations into daily features, including average temperature, maximum temperature, and total daily precipitation. This required parsing timestamps, creating daily groupings, and computing summary statistics.

2) Economic Indicators Integration (FRED API):

Using the fredr package, I fetched multiple macroeconomic indicators with varying frequencies (daily interest rates, weekly mortgage rates, monthly unemployment and CPI, and quarterly GDP and public debt). I reshaped the data into wide format and constructed a continuous daily timeline. To address mixed frequencies, I applied a Last Observation Carried Forward (LOCF) strategy so that lower-frequency variables could be aligned to daily observations.

3) News-Based Signals (GDELT API):

I experimented with GDELT's document API to retrieve timeline-based sentiment data for specific themes (e.g., protests, unrest, natural disasters) filtered by source country. This establishes a foundation for incorporating event-driven or sentiment-based external signals into the analysis.

Role in the Larger Project

This work contributes directly to our group objective of building an R package that automates the attachment of macro-level contextual data to user-provided revenue datasets. The scripts developed today form the technical backbone for:

1. collecting heterogeneous external data,
2. standardizing them to a common daily time scale,
3. and preparing them for downstream visualization and trend analysis.

These components will later be wrapped into higher-level functions so analysts can enrich their revenue data with macro features using a single function call.

Development Decisions

1. I chose Open-Meteo for weather data because it is free, does not require authentication, and provides hourly resolution suitable for daily aggregation.
2. I used FRED for economic indicators due to its reliability and wide coverage of macroeconomic variables.
3. For mixed-frequency economic data, I implemented Last Observation Carried Forward (LOCF) to align all series to daily granularity, as this reflects how macro indicators are typically treated in applied analytics.
4. I explored GDELT for news-based signals since it offers theme-based queries and timeline sentiment, making it suitable for capturing societal disruptions or major events.
5. I initially relied on base R aggregation functions to prototype quickly, with the intention of later refactoring into cleaner, reusable package functions.

```
# load packages

library(tidyverse)
```

```
## Warning: package 'readr' was built under R version 4.5.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.6
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.1.0
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(fredr)
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 4.5.2
```

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

```
## ----------------------------------------  WEATHER API ----------------------------------------

lat <- 51.5074 # location lattitude
lon <- -0.1278 # location longitude
start <- "2026-01-01" # start date
end   <- "2026-01-25" # end date

# 1. CALL API
url <- paste0("https://api.open-meteo.com/v1/forecast?",
              "latitude=", lat, "&longitude=", lon,
              "&start_date=", start, "&end_date=", end,
              "&hourly=temperature_2m,precipitation",
```

```
              "&timezone=auto",
              "&format=csv")

# 2. Read and Clean Data
weather_data <- read.csv(url, skip = 10, header = FALSE)
colnames(weather_data) <- c("datetime", "temp_c", "precip_mm")
weather_data$datetime <- as.POSIXct(weather_data$datetime, format="%Y-%m-%dT%H:%M")
weather_data$date <- as.Date(weather_data$datetime)

# 4. Calculate stats
daily_summary <- aggregate(cbind(temp_c, precip_mm) ~ date,
                           data = weather_data,
                           FUN = function(x) c(mean = mean(x), max = max(x), sum = sum(x)))

# 5. Clean up the messy matrix output from aggregate
daily_summary <- data.frame(
  date      = daily_summary$date,
  avg_temp  = round(daily_summary$temp_c[, "mean"], 1),
  max_temp  = daily_summary$temp_c[, "max"],
  tot_rain  = daily_summary$precip_mm[, "sum"]
)

print(daily_summary)
```

```
##          date avg_temp max_temp tot_rain
## 1  2026-01-01      3.5      6.2      0.0
## 2  2026-01-02      3.4      6.1      0.3
## 3  2026-01-03      1.2      3.4      0.0
## 4  2026-01-04     -0.1      2.5      0.0
## 5  2026-01-05      0.3      1.8      0.0
## 6  2026-01-06     -0.6      1.8      0.1
## 7  2026-01-07      3.7      6.3      8.0
## 8  2026-01-08      3.9      6.1      1.7
## 9  2026-01-09      5.1      8.8     12.6
## 10 2026-01-10      2.9      4.7      0.0
## 11 2026-01-11      2.8      7.3      0.0
## 12 2026-01-12     10.0     11.8      1.6
## 13 2026-01-13     10.2     11.2      8.3
## 14 2026-01-14      5.7     10.3      2.4
## 15 2026-01-15      8.9     10.4      8.3
## 16 2026-01-16      7.6     10.0      8.1
## 17 2026-01-17      8.5     10.9      0.3
## 18 2026-01-18      8.3     11.7      0.9
## 19 2026-01-19      9.3     11.4      0.0
## 20 2026-01-20      9.7     10.7      0.8
## 21 2026-01-21      8.6     10.0      7.0
## 22 2026-01-22      9.2     10.0      5.5
## 23 2026-01-23      8.3      9.5      3.0
## 24 2026-01-24      7.7      9.6      0.0
## 25 2026-01-25      7.7      8.8      2.2
## 26 2026-01-26      6.2      7.1      0.0
## ---------------------------------------  ECONOMIC API  ----------------------------------------
```

```r
# Set API key

fred_api_key <- '1f52a3bcb927a3a2423a9a2e78bd1261'
fredr_set_key(fred_api_key)

# 1. Define the parameters and their frequencies
indicators <- c(
  "Interest_Rate"     = "DFF",          # Daily
  "Mortgage_Rate"     = "MORTGAGE30US", # Weekly
  "Unemployment"      = "UNRATE",       # Monthly
  "CPI_Inflation"     = "CPIAUCSL",     # Monthly
  "GDP"               = "GDPC1",        # Quarterly
  "Public_Debt"       = "GFDEBTN"       # Quarterly
)

# 2. Fetch all data at once
raw_data <- map_dfr(indicators,
                    ~fredr(series_id = .x, observation_start = as.Date("2020-01-01")),
                    .id = "metric")

# 3. Pivot to Wide Format
daily_trend <- raw_data %>%
  select(date, metric, value) %>%
  pivot_wider(names_from = metric, values_from = value) %>%
  arrange(date)

# 4. The "Fill" Step (LOCF)
full_dates <- data.frame(date = seq(min(daily_trend$date), max(daily_trend$date), by="day"))

final_data <- full_dates %>%
  left_join(daily_trend, by = "date") %>%
  mutate(across(-date, ~na.locf(., na.rm = FALSE)))

# View the result
head(final_data)
```

```
##         date Interest_Rate Mortgage_Rate Unemployment CPI_Inflation      GDP
## 1 2020-01-01          1.55            NA          3.6       259.127 20709.21
## 2 2020-01-02          1.55          3.72          3.6       259.127 20709.21
## 3 2020-01-03          1.55          3.72          3.6       259.127 20709.21
## 4 2020-01-04          1.55          3.72          3.6       259.127 20709.21
## 5 2020-01-05          1.55          3.72          3.6       259.127 20709.21
## 6 2020-01-06          1.55          3.72          3.6       259.127 20709.21
##   Public_Debt
## 1    23223813
## 2    23223813
## 3    23223813
## 4    23223813
## 5    23223813
## 6    23223813
```

```r
tail(final_data)
```

```
##            date Interest_Rate Mortgage_Rate Unemployment CPI_Inflation      GDP
## 2209 2026-01-17          3.64          6.06          4.4        326.03 24026.83
```

```
## 2210 2026-01-18          3.64          6.06          4.4        326.03 24026.83
## 2211 2026-01-19          3.64          6.06          4.4        326.03 24026.83
## 2212 2026-01-20          3.64          6.06          4.4        326.03 24026.83
## 2213 2026-01-21          3.64          6.06          4.4        326.03 24026.83
## 2214 2026-01-22          3.64          6.09          4.4        326.03 24026.83
##      Public_Debt
## 2209    37637553
## 2210    37637553
## 2211    37637553
## 2212    37637553
## 2213    37637553
## 2214    37637553
```

```
## ----------------------------------------  NEWS API  ----------------------------------------

# ----------------------------------------  POLITICAL UNREST ----------------------------------------

# 1. Define QUERY
query_text <- '(protest OR unrest) sourcecountry:Canada'

# 2. Call API
url <- paste0("https://api.gdeltproject.org/api/v2/doc/doc?",
              "query=", URLencode(query_text),
              "&mode=TimelineTone",
              "&timespan=1m",
              "&format=CSV")

# 3. Read the data
raw_data <- read.csv(url, check.names = FALSE)

raw_data
```

```
##          Date       Series   Value
## 1  2025-12-28 Average Tone -2.5670
## 2  2025-12-29 Average Tone -3.4264
## 3  2025-12-30 Average Tone -4.3969
## 4  2025-12-31 Average Tone -3.0045
## 5  2026-01-01 Average Tone -3.8409
## 6  2026-01-02 Average Tone -2.9814
## 7  2026-01-03 Average Tone -2.3810
## 8  2026-01-04 Average Tone -3.1598
## 9  2026-01-05 Average Tone -2.7636
## 10 2026-01-06 Average Tone -3.0106
## 11 2026-01-07 Average Tone -4.1565
## 12 2026-01-08 Average Tone -2.9174
## 13 2026-01-09 Average Tone -5.9883
## 14 2026-01-10 Average Tone -3.5328
## 15 2026-01-11 Average Tone -3.7314
## 16 2026-01-12 Average Tone -4.5460
## 17 2026-01-13 Average Tone -4.6480
## 18 2026-01-14 Average Tone -3.5609
## 19 2026-01-15 Average Tone -2.8523
## 20 2026-01-16 Average Tone -2.6826
## 21 2026-01-17 Average Tone -3.5737
## 22 2026-01-18 Average Tone -2.9561
```

```
## 23 2026-01-19 Average Tone -1.9715
## 24 2026-01-20 Average Tone -2.9744
## 25 2026-01-21 Average Tone -3.0106
## 26 2026-01-22 Average Tone -2.9674
## 27 2026-01-23 Average Tone -2.5792
## 28 2026-01-24 Average Tone -2.9797
## 29 2026-01-25 Average Tone -3.4008
## 30 2026-01-26 Average Tone -3.9406
```

```r
# ---------------------------------------  NATURAL DISASTERS ----------------------------------------

# 1. Define QUERY
query_text <- '(theme:NATURAL_DISASTER OR "disaster" OR "storm" OR "flood") sourcecountry:CA'

# 2. Call API
url <- paste0("https://api.gdeltproject.org/api/v2/doc/doc?",
              "query=", URLencode(query_text),
              "&mode=TimelineTone",
              "&timespan=1m",
              "&format=CSV")

# 3. Read the data
raw_data <- try(read.csv(url, check.names = FALSE), silent = TRUE)

raw_data
```

```
##          Date       Series   Value
## 1  2025-12-28 Average Tone -2.9536
## 2  2025-12-29 Average Tone -2.2224
## 3  2025-12-30 Average Tone -1.7763
## 4  2025-12-31 Average Tone -1.7513
## 5  2026-01-01 Average Tone -1.7308
## 6  2026-01-02 Average Tone -0.0120
## 7  2026-01-03 Average Tone  0.5627
## 8  2026-01-04 Average Tone -0.8838
## 9  2026-01-05 Average Tone -1.0725
## 10 2026-01-06 Average Tone -1.4155
## 11 2026-01-07 Average Tone -1.0040
## 12 2026-01-08 Average Tone -1.2359
## 13 2026-01-09 Average Tone -1.6009
## 14 2026-01-10 Average Tone -1.0470
## 15 2026-01-11 Average Tone -1.2991
## 16 2026-01-12 Average Tone -1.3889
## 17 2026-01-13 Average Tone -1.3502
## 18 2026-01-14 Average Tone -1.6203
## 19 2026-01-15 Average Tone -0.9429
## 20 2026-01-16 Average Tone -1.2232
## 21 2026-01-17 Average Tone -0.7790
## 22 2026-01-18 Average Tone -0.5287
## 23 2026-01-19 Average Tone -2.3838
## 24 2026-01-20 Average Tone -0.8746
## 25 2026-01-21 Average Tone -0.7731
## 26 2026-01-22 Average Tone -1.5193
## 27 2026-01-23 Average Tone -2.0712
## 28 2026-01-24 Average Tone -1.8772
```

```
## 29 2026-01-25 Average Tone -2.1312
## 30 2026-01-26 Average Tone -1.7078
```

Commit link: https://github.com/mzikkhan/a.u.r.o.r.a/commit/42e75ed3286d08483fc0b61e00bb52fd46501934#diff-e90f1dd576380e55c0157d832b265fcec51a1d050cdf6168a20ae0779a4e9e88R2-R4

26-Jan-2026