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.

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

26-Jan-2026

Summary

Today I focused on productionizing the macro-level data integration prototype by refactoring the codebase into a structured R package and improving reliability, usability, and maintainability.

I completed seven major tasks:

1) Code Documentation and Readability

I added comprehensive inline comments across all core scripts, clearly explaining API calls, data transformations, aggregation logic, and alignment steps. This improves onboarding for future contributors and makes the pipeline easier to debug and extend.

I also standardized naming conventions, variable scopes, and function signatures to improve overall code clarity.

2) Code Restructuring and Cleanup

I refactored the initial prototype into modular, reusable functions:

a) Separated API logic from transformation logic

b) Removed duplicated blocks

c) Centralized configuration parameters

d) Simplified aggregation pipelines

This resulted in cleaner abstractions and reduced technical debt, making each component independently testable.

3) Unified Macro Data Function

I packaged all three external data sources (Open-Meteo, FRED, and GDELT) into a single high-level function that:

a) accepts user inputs (location, date range, indicator selections)

b) fetches weather, economic, and news-based signals

c) standardizes all outputs to a daily time series

d) applies LOCF for mixed-frequency macro variables

e) and returns a merged dataframe ready to be joined with revenue or performance datasets

This creates a one-call interface from the main script, significantly simplifying analyst workflows.

4) Professional R Package Structure

I reorganized files to follow standard R package conventions:

a) R/ for core functions

b) tests/testthat/ for unit and integration tests

c) man/ for generated documentation

d) DESCRIPTION and NAMESPACE for dependency and export management

e) inst/ for example data and auxiliary resources

This establishes a production-ready foundation aligned with CRAN-style best practices.

5) Unit and Integration Testing

I implemented:

a) Unit tests for each API wrapper and transformation function

b) Integration tests validating end-to-end pipeline execution

Tests cover:

a) successful API responses

b) malformed or empty inputs

c) mixed-frequency alignment

d) merged output schema

This ensures correctness and guards against regressions as features evolve.

6) Error Handling and Robustness

I added tryCatch() blocks around all external API calls and critical transformations to gracefully handle:

a) network failures

b) missing or malformed responses

c) unexpected schema changes

Meaningful error messages are returned to users, preventing silent failures and improving debuggability.

7) Functional Documentation

I documented all exported functions using roxygen2-style comments, including:

a) parameter definitions

b) return values

c) usage examples

d) edge-case behavior

This enables automatic generation of help files and provides clear guidance for downstream users.

Role in the Larger Project

These improvements transition the pipeline from an exploratory prototype into a reusable analytics tool. The package now supports:

a) automated macro-data enrichment,

b) robust error handling,

c) reproducible workflows,

d) extensible architecture.

This directly advances our group objective of delivering a professional R package that allows analysts to attach macro-level contextual signals to revenue datasets with a single function call.

Development Decisions

a) Modularized API logic to isolate external dependencies.

b) Centralized the three data sources behind a unified interface for simplicity.

c) Adopted LOCF as the default alignment strategy for economic indicators.

d) Followed standard R package structure to support long-term maintenance.

e) Prioritized testing and error handling to improve production readiness.

f) Used roxygen2 for documentation to enable automatic help-file generation.

Commit links:

https://github.com/mzikkhan/a.u.r.o.r.a/commit/0dc4a70a9f0346a98a2f7747691aa8fd7dcb03b4

https://github.com/mzikkhan/a.u.r.o.r.a/commit/f79134d36ebcd72da860036e023d9873c9421182

https://github.com/mzikkhan/a.u.r.o.r.a/commit/95f63000dee06889c2eba38752980674d10e29c1

https://github.com/mzikkhan/a.u.r.o.r.a/commit/e52f2eda1ca692d43891a44c285f789f76f48237