Project — Code & Report

Finance Project — Asset Management

Prof. Dr. Andre Guettler Oliver Padmaperuma

Overview

> **Due: 30 June 2026, 18:00** (firm) via email. **Submission format:** one zip-folder containing your Rmd + report PDF + slide PDF. **Group size: 3** students (we allocate if you don't form one). **Weight: 50%** of the final grade. The remaining 50% is graded on the live presentation (1 July 2026) — see the presentation brief.

Build an end-to-end empirical asset-management research pipeline in **R**, applied to the **Polymarket Quant Bench** dataset (curated OHLCV bars over Jon Becker's polymarket-data on-chain dump). You will engineer a small library of indicators, combine them into a trading signal, back-test the strategy, and write a critical reflection.

Submission rules

You submit **a single zip-folder** named — and email subject line —

`Asset2026_surname1_surname2_surname3`

The zip must contain exactly **three files**:

1. **`<groupname>.Rmd`** — your R Markdown source. Self-explanatory, well-commented, vectorised; helper functions for repetitive logic with one-line docstrings. Hard-coded paths are not permitted (use a relative `data/` directory).
2. **`<groupname>_report.pdf`** — the PDF knitted from the Rmd. 10–15 pages including figures and tables.
3. **`<groupname>_slides.pdf`** — your final-presentation slides as PDF (no PowerPoint binaries).

Email the zip to oliver.padmaperuma@uni-ulm.de and put andre.guettler@uni-ulm.de **as well as your team-mates** in CC. If the attachment exceeds the mail server's limit, share a cloud link instead.

Dataset

- **Source:** Polymarket Quant Bench — OHLCV bars over Jon Becker's polymarket-data raw dump. Public, CC-BY-4.0.
- **Access — one-time CLI download, then R reads the local copy.** Set up your project folder like this: `asset-group-NN/ asset-group-NN.Rproj asset-group-NN.Rmd data/ ← create this; add to .gitignore polymarket/ ← dataset lands here .gitignore` Then in a terminal (PowerShell / Terminal / Anaconda Prompt), **once per machine**: `bash pip install huggingface_hub hf download smf-ulm/polymarket-quant-bench \ --repo-type dataset \ --local-dir data/ # repo's top-level polymarket/ lands here` The CLI shows a live progress bar; the download is ~603 MB and resumable. **Pin `--revision <sha>` to a specific HuggingFace commit before you submit** so the marker re-runs against your snapshot.

  Troubleshooting if the install gets stuck:

  - **No Python** → install from https://python.org (tick "Add Python to PATH" on Windows). `pip` ships with Python 3.
  - **`pip` not recognised** → `py -m pip install huggingface_hub` (Windows) or `python3 -m pip install huggingface_hub` (macOS/Linux).
  - **`hf` not recognised after install** → restart your terminal; if still missing, use `python -m huggingface_hub download …` with the same flags.
  - **Saw `huggingface-cli is deprecated, use hf instead`** → just replace `huggingface-cli` with `hf`. Same flags.
  - **Download interrupted** → just re-run the same `hf download …` command (it resumes).
  - **Behind a proxy** → set `HTTPS_PROXY=http://<proxy>:<port>` before running the CLI.

  After the download, in your Rmd: `r library(arrow) local_path <- "data/polymarket" markets <- open_dataset(file.path(local_path, "markets")) |> collect() bars_daily <- open_dataset(file.path(local_path, "bars_daily")) |> collect() bars_hourly <- open_dataset(file.path(local_path, "bars_hourly")) |> collect()`
- **Three configs**:
  - **`markets`** (36,831 rows) — one row per resolved Polymarket market with `$100k` cumulative volume and `200` trade fills. Columns: `id`, `condition_id`, `question`, `slug`, `category`, `outcomes`, `outcome_prices` (JSON), `clob_token_ids` (JSON `[yes, no]`), `volume`, `liquidity`, `created_at`, `end_date`.

  - `bars_daily` (~1.46M rows) — one row per (`token_id`, calendar day). Columns: `token_id`, `period_start`, `period_end`, `open`, `high`, `low`, `close`, `vwap`, `volume_usd`, `n_trades`, `n_buys`, `n_sells`. **Prices are implied probabilities in `[0, 1]`.**
  - `bars_hourly` (~12.66M rows) — identical schema, hourly resolution.

- **Important conventions**: bars are per **token** (YES *and* NO are separate series — pair via `clob_token_ids`). Bars are **sparse** (no row for periods with zero trades — forward-fill with `tidyr::fill()`). Timestamps are **UTC**. Credit Jon Becker's `polymarket-data` as the upstream source whenever you reference the data.

Exercise 1 — Prepare your data

Step	Task
a) Load	Use `arrow::open_dataset() |> collect()` on each of the three configs (`markets/`, `bars_daily/`, `bars_hourly/`) to materialise them as regular tibbles. Inspect with `glimpse()` and `summary()`.
b) Define your universe	The dataset is **pre-filtered** to liquid markets ($100k volume, 200 fills). Further narrow with **clear inclusion rules**: category (Politics / Sports / Crypto / …), `created_at` / `end_date` window, minimum number of `bars_daily` rows per market, possibly excluding parent / child siblings via `condition_id`. Justify each rule in the report.
c) Clean & transform	Forward-fill sparse bars within each `token_id` with `tidyr::fill(close, .direction = "down")` after `arrange(period_start)`. Parse `outcome_prices` / `clob_token_ids` from JSON via `jsonlite::fromJSON()`. Filter `bars_daily` to one side (YES) using `clob_token_ids[1]` unless you build a market-mid. Save as `bars_clean`.

Step	Task
d) Descriptive analysis	Per category, report median lifetime (`end_date - created_at`), median volume, **Yes-win-rate** (parse `outcome_prices` JSON — the winner closes near 1.0), and the cross-sectional distribution of terminal close prices. Build at least **one summary table** (with `kableExtra`) and **one figure** (with `ggplot2`).

Exercise 2 — Engineer your indicator library

Build **at least 5–7 indicators** drawn from the buckets below. **For each indicator state**: (i) the formula, (ii) the economic / behavioural intuition, (iii) the rolling window or hyper-parameter, and (iv) any external data dependency.

- **Trend** — SMA / EMA crossovers, MACD, slope of a rolling linear fit.
- **Momentum** — RSI, k-day return, return percentile within a rolling window.
- **Volatility** — Bollinger bands, rolling std-dev, range-based volatility (Parkinson / Garman-Klass).
- **Volume / liquidity** — VWAP, volume z-score, bid-ask spread (when available).
- **Time-to-resolution** — days-to-resolution, log-clock decay, calendar-effect indicators (weekday, month).
- **Cross-market** — correlation with related Polymarket markets, parent / child contracts.
- **External signals (optional but encouraged)** — Google Trends (`gtrendsR`), news sentiment (`tidytext`), polls, sports / political odds, weather.

  Avoid `for` loops where possible — use `slider::slide_dbl`, `dplyr::mutate(across())`, or `data.table`. Marked down at grading.

Exercise 3 — Combine into a signal & back-test

Step	Task
a) Combine	Use Ridge / Lasso / Elastic Net (`glmnet`) to combine your indicators into a single signal. Tune λ (and α if Elastic Net) by **K-fold CV on the training data only**.
b) Walk-forward	Run a strict walk-forward back-test — re-fit on data through $t-1$, predict at t, advance one step. **Never tune on the test fold.**

Step	Task
c) Trading rule	Define a clear, simple trading rule that maps the predicted signal to a position. State assumed transaction costs explicitly (a flat 1 % per trade is honest; ignoring costs is not).
d) Performance	Report at minimum: cumulative P&L, hit rate, average return per trade, a Sharpe-like metric. Use `PerformanceAnalytics` where helpful.
e) Robustness	Repeat on a **held-out cohort of markets** (e.g. markets that resolved in the last 20 % of the sample window). Does the strategy still work?

Exercise 4 — Reflect & write the report

Sections (10–15 pages, R Markdown → PDF):

Section	Min. length	Content
Introduction	0.5 page	What you set out to test and why prediction markets are an interesting test bed.
Data	1.5 pages	Source, snapshot version, your universe rules, cleaning steps, headline statistics.
Indicators	2 pages	Each indicator: formula, intuition, hyper-parameter. One table summarising them.
Methodology	1.5 pages	Estimator (Ridge / Lasso / EN), CV scheme, walk-forward design, trading rule, transaction-cost assumption.
Results	3 pages	Back-test performance, robustness on a held-out cohort, at least **two figures** and **two tables**.

Section	Min. length	Content
Discussion	1.5 pages	What works, what doesn't, where prediction markets break technical-analysis intuitions, what you would do next.
Conclusion	0.5 page	Three-sentence takeaway.

You may use an LLM (e.g. ChatGPT) ***to refine*** *prose and check derivations — not to generate the substantive content. Disclose any AI assistance in a footnote.*

Recommended R packages

Category	Packages
Data access	`arrow` (after the one-time `hf download …` — see *Dataset* above)
Data wrangling	`tidyverse`, `lubridate`, `data.table`
Time-series core	`xts`, `zoo`, `tsibble`, `slider`
Technical indicators	`TTR`, `quantmod`, `tidyquant`
ML / regularisation	`glmnet`, `caret` or `tidymodels`
Back-test / metrics	`PerformanceAnalytics`
Plotting / reporting	`ggplot2`, `patchwork` (panel composition), `rmarkdown`, `knitr`, `kableExtra`
Optional external data	`gtrendsR` (Trends), `tidytext` (sentiment), `gh` (GitHub API)

Grading rubric

Criterion	Weight
Code quality (efficiency, vectorisation, comments, naming, helper functions)	25%
Creativity (indicator menu, external-data integration, robustness ideas)	20%
Empirical correctness (CV discipline, walk-forward integrity, transaction costs)	25%
Writing (concise, economically motivated, skim-friendly tables and plots)	20%

Criterion	Weight
Reproducibility (the Rmd actually runs end-to-end on the dataset we shipped)	10%

Honor code

By submitting this project, you confirm that the work is your group's own, that all sources are cited, and that any AI assistance has been disclosed.