Futures Roll Analysis Framework

Copper (HG) Case Study 2008–2024

Abstract

I present a unified Python framework for detecting institutional roll activity in futures markets. Minute-level CME copper (HG) contracts (2008–2024) are ingested, aggregated into variable-granularity buckets, and analysed using a vectorised calendar-spread detection engine. The refactored codebase introduces a conventional `src/` package layout, reusable services, and command-line entry points that reproduce key empirical findings: 2,736 hourly roll events (6.16% hit-rate) with a median timing of 28 days before expiry, and 183 daily events after contract-quality filtering with a 14-day median lead.

1 Framework Overview

1.1 Objectives

- Provide a reproducible pipeline for futures roll-detection research.
- Standardise data ingest, aggregation, panel assembly, and event detection across commodities.
- Surface empirical roll statistics (timing, session distribution, liquidity confirmation).

1.2 Repository Layout

```
src/futures_roll_analysis/
  analysis.py        # hourly & daily orchestration services
  buckets.py         # 10-session schema and aggregation helpers
  ingest.py          # minute-file discovery and contract aggregation
  panel.py           # wide panel assembly with expiry metadata
  rolls.py           # vectorised front/next, spreads, liquidity signals
  events.py          # spread-widening detection & summaries
  quality.py         # data-quality filtering and reporting utilities
  cli/               # command-line entry points (hourly, daily, organise)
```

Configuration and metadata

- `config/settings.yaml`: Source-of-truth for commodity root, timezone, bucket detection parameters, and quality thresholds.
- `metadata/contracts_metadata.csv`: Official CME expiry dates used for front/next identification.

Outputs

- `outputs/panels/`: Parquet & CSV panels (hourly and filtered daily).
- `outputs/roll_signals/`: Spread, widening, and liquidity series.
- `outputs/analysis/`: Summary tables (bucket distribution, preference scores, transition matrix, daily widening summary).
- `outputs/data_quality/`: Contract-level metrics and JSON summary produced during daily runs with filtering enabled.

2 Data Preparation

2.1 Source and Scope

- Exchange: CME Group copper (HG) futures.
- Horizon: January 2008–December 2024 (202 listed contracts).
- Frequency: 1-minute OHLCV with optional open-interest fields.
- Files: 13,548 source files spanning 32 commodities, organised into 32 folders via the CLI organiser.
- Timezone: US/Central (Chicago).

2.2 Organising Raw Data

Raw `*.txt` files are placed under asset-specific directories via:

```
python -m futures_roll_analysis.cli.organize \
  --source /path/to/raw_txt \
  --destination organized_data
```

The organiser recognises contract symbols, creates commodity folders, and writes `data_inventory.csv`.

2.3 Data Filtering

- Contracts failing minimum density, coverage, or gap constraints are excluded.
- Thresholds configurable in `settings.yaml` (default: 500 data points, coverage $\geq$25%, max gap 30 days, cutoff year 2015).
- Reports saved as CSV and JSON in `outputs/data_quality`.

3 Methodology

3.1 Minute Aggregation to Buckets

1. Localise/convert minute timestamps to US/Central.
2. Assign each minute to one of 10 sessions (7 one-hour US periods, plus Late US, Asia, Europe).
3. Anchor buckets by start timestamp (e.g., Asia session begins 21:00 of prior day).
4. Aggregate OHLCV per bucket (open=first, close=last, high=max, low=min, volume=sum).

Vectorised helpers in `buckets.py` avoid per-row Python loops; validation utilities ensure volume/price conservation.

3.2 Panel Assembly

- Each contract receives a dedicated column block (open, high, low, close, volume, expiry metadata).
- Metadata bucket columns are merged into a single `meta` namespace to avoid duplication.
- Expiry dates are sourced from CME metadata and normalised to midnight.

3.3 Front/Next Identification and Spread Calculation

- `rolls.identify_front_next`: vectorised computation using expiry arrays and `numpy.argmin`.
- Active contracts require: (a) finite price, (b) expiry date $\geq$ observation date.
- Spread = next close − front close; liquidity roll = next volume $\geq \alpha \cdot$ front volume (default $\alpha = 0.8$).

3.4 Spread-Widening Detection

- Change series: $\Delta s_t = s_t - s_{t-1}$.
- Z-score window: 20 buckets (approx. 2 trading days) for hourly pipeline; 30 days for daily pipeline.
- Threshold: $z > 1.5$ plus optional absolute floor (unused by default).
- Cool-down: 3-hour time delta for buckets or configurable integer window for daily analysis.
- Outputs: boolean widening series, bucket-level summaries, preference scores (event rate vs. volume share), transition matrix.

4 Empirical Results (HG)

4.1 Hourly Bucket Insights

- Total buckets: 44,428 (2008–2024).
- Widening events detected: 2,736 (frequency 6.16%).
- Median timing: 28 days before expiry (IQR 16–43, range 0–116).

Table 1: Roll Events by Intraday Period (2008–2024)

Bucket	Session Label	Events	Share (%)	Pref. Score
1	09:00 – US Open	383	14.00	1.04
2	10:00 – US Morning	284	10.38	0.73
3	11:00 – US Late Morning	238	8.70	0.70
4	12:00 – US Midday	186	6.80	0.60
5	13:00 – US Early Afternoon	132	4.82	0.87
6	14:00 – US Late Afternoon	198	7.24	**2.18**
7	15:00 – US Close	151	5.48	**2.02**
8	Late US / After-Hours	296	10.78	0.87
9	Asia Session	454	16.67	0.47
10	Europe Session	414	15.13	0.51

Observation US regular hours account for 57.4% of events despite fewer minutes per day and dominate the preference scores—late afternoon and close sessions exhibit more than twice the expected activity based on volume share, highlighting concentrated institutional rolling near settlement.

4.2 Daily Panel Validation

- Contracts analysed after filtering: 109 (out of 202).
- Valid spread observations: 3,309 across 5,105 trading days.
- Widening events detected: 183 (3.6% hit-rate).
- Median days-to-expiry: 14 (mean 24.0, range 0–352); daily view emphasises late rollers once sparse contracts are excluded.
- Liquidity rolls (volume-based) confirm 64% of spread signals.

5 Usage Guide

5.1 Installation

```
python -m pip install --upgrade pip
python -m pip install -e .[dev,viz]
```

The editable install exposes console scripts:

- `futures-roll-organize`
- `futures-roll-hourly`
- `futures-roll-daily`

5.2 Hourly Analysis

```
futures-roll-hourly \
  --settings config/settings.yaml \
  --root organized_data/copper \
  --output-dir outputs
```

Optional flags: `--metadata`, `--max-files`, `--log-level`.

5.3 Daily Analysis

```
futures-roll-daily \
  --settings config/settings.yaml \
  --root organized_data/copper \
  --output-dir outputs
```

5.4 Testing

```
pytest
```

Unit tests cover bucket assignment/aggregation, Z-score detection, and cool-down behaviour.

6 Conclusion and Future Work

6.1 Key Takeaways

- Refactored architecture cleanly separates ingest, aggregation, event detection, and CLI orchestration.
- Hourly analytics reveal heterogeneous roll behaviour, with statistically significant clustering during 14:00–15:00 CT.
- Daily validation (with quality filters) confirms persistent signals and highlights the importance of contract selection.