TCGBERG · ANALYTICS METHODOLOGY

Fair Value Methodology

An adaptive, sale-based fair value engine for thinly-traded heterogeneous assets with built-in confidence scoring.

Version 1.0 | May 2026
Status: Specification — ready for implementation
Author: CK
Scope: per-printing, per-grader, per-grade, per-date

PROBLEM STATEMENT

1. Why Card Pricing Is Hard

Trading cards are not equities. They do not trade continuously, prices are heterogeneous across condition tiers and grading services, and individual liquidity profiles vary by orders of magnitude. A standard rolling-window moving average — the default approach for liquid asset pricing — fails badly here.

A specific Charizard PSA 10 might trade five times in a week. Its raw 1st Edition counterpart might trade once a month. A common card from the same set might not have traded for six months. Yet for the analytical layer to work, every printing-grade pair needs a defensible single value.

Standard Approaches and Their Failure Modes

Approach	Failure Mode
N-day rolling median	Returns NULL for thin cards. Breaks the analytical pipeline.
Last single sale	Single-sample noise. One bad fill distorts the value.
Time-weighted EWMA	Ambiguous when sales are sparse over uneven intervals.
Average of last N sales	No outlier defense. Shill bids and errors dominate small samples.
Drop outliers, then average	Discards information unnecessarily. Bad for small N.

Our framework addresses each of these by combining four estimation methods, weighting them adaptively, winsorizing rather than dropping outliers, and producing a numeric confidence score that downstream consumers (indices, deal screener, grading ROI) can use to decide how much to trust the output.

DESIGN PRINCIPLES

2. What the Algorithm Optimizes For

- **Sale-based, not time-based.** The algorithm asks "what were the last N sales?" not "what happened in the last N days?" This matches how dealers, flippers, and graders actually think about value.
- **Always produces a value when any data exists.** Even a single recent sale produces a fair value, with appropriately low confidence. The pipeline never returns NULL except when no sales exist at all.
- **Confidence is a continuous score, not a label.** A 0-100 numeric score allows fine-grained downstream filtering. "Include in indices only if confidence ≥ 60" is more useful than three buckets.
- **Winsorize, don't drop.** When an outlier is detected, its price is clipped to the 1st/99th percentile rather than removed. Preserves sample size while neutralizing distortion.
- **Adaptive method blending.** The same card may need different methods at different times. Strong trend → lean on EWMA. Tight cluster → lean on median. High recent density → lean on recent sales.
- **Full transparency.** Every fair value is persisted with its method blend, sub-scores, and diagnostics. Anyone can audit "why does this card score 42?"

ALGORITHM

3. Algorithm Overview

The algorithm proceeds through nine stages. Each stage is deterministic and unit-testable. All thresholds are stored in the `config` table for tuning without code changes.

Stage	Purpose
1. Gather sample	Pull last 30 sales, convert all prices to USD
2. Winsorize	Clip prices below p01 / above p99 (only if n ≥ 5)
3. Compute diagnostics	Sample stats: counts, recency, density, dispersion, trend
4. Compute method outputs	Four candidate point estimates
5. Determine adaptive blend weights	Apply rules based on diagnostics
6. Compute point estimate	Weighted average of method outputs
7. Compute confidence sub-scores	Five 0-100 sub-scores
8. Compute final confidence score	Weighted sum, rounded to integer
9. Bucket and persist	Categorize, write to fair_values table

STAGE 1

4. Gather Sample

For each (printing, grader, grade) tuple on a given as-of-date:

- Pull all `raw_prices` rows where `printing_id`, `grader_id`, and `grade_id` match, with `price_date <= as_of_date`.
- Order by `price_date` descending (most recent first).
- Convert each price to USD using hardcoded FX rates: USD = 1.0, EUR = 1.08, GBP = 1.27, JPY = 0.0067. (FX ingestor is a separate future ticket.)
- Take the first 30 entries — sample is bounded for performance and to avoid stale-history contamination.
- If the result is empty: return a row with all NULLs and confidence 0.

STAGE 2

5. Winsorization

Winsorization replaces extreme values with the corresponding percentile cutoff, rather than removing them entirely. This preserves sample size — critical for thin-traded cards where every observation matters — while neutralizing the distortion of erroneous transactions, shill bids, and charity sales.

If `n_total < 5`: skip winsorization (sample too small for reliable percentiles).

Otherwise:

- Compute the 1st percentile (*p01*) and 99th percentile (*p99*) of sample prices.
- For each sale where `price < p01`: replace its price with `p01`.
- For each sale where `price > p99`: replace its price with `p99`.
- The sale's date and other metadata remain unchanged.

> Why winsorize at 1/99 instead of 5/95? With samples up to 30, the 1st/99th percentile only touches the actual extremes — a 5/95 cut would aggressively trim normal variance. The 1/99 cut only intervenes when something is genuinely anomalous.

STAGE 3

6. Diagnostics

Six independent diagnostic dimensions characterize the sample. These drive both the method blend weights and the confidence score.

Diagnostic	Definition	What It Measures
n_total	Count of sales in winsorized sample	Sample adequacy
last_sale_date	Date of most recent sale	Recency

Diagnostic	Definition	What It Measures
days_since_last_sale	as_of_date – last_sale_date	Recency
n_last_30d / 90d / 180d / 365d	Sale counts in time windows	Sales pace context
mean_gap_days	Mean gap between consecutive sale dates	Sales density / liquidity
price_cov	stdev / mean of raw (pre-winsor) prices	Price stability
trend_slope, trend_r²	OLS of ln(price) on days_ago, last 20 sales	Directional drift
has_outliers	Were any prices winsorized?	Sample cleanliness

Note: `price_cov` is computed on the *raw* (pre-winsor) prices. This captures the true variance in the underlying market, not the post-winsor sample.

STAGE 4

7. The Four Methods

The point estimate is a weighted blend of up to four candidate values. Each method has a distinct strength.

7.1 Method A — Sale-Rank EWMA (last 10 sales)

Exponentially weighted moving average where the weight depends on the sale's rank, not its calendar date. With halflife of 3 sales, the most recent sale has weight 1.0, the next has 0.79, then 0.63, 0.50 (halfway), 0.40, and so on.

```
weight(rank) = exp( -ln(2) × rank / halflife_n )
method_a = Σ (weight × price) / Σ weight
```

Best for: liquid cards with consistent recent sales. Captures momentum without being too jumpy.

7.2 Method B — Trimmed Median (last 10 sales)

Median of the last 10 sales (post-winsorization). Since winsorization already neutralized extremes, no further trimming is applied.

Best for: cards with high price variance where the central tendency matters more than recency. Robust to remaining noise after winsorization.

7.3 Method C — Recent-Window Robust (last 30 days)

Median of all sales within the last 30 calendar days. Only computed if n_last_30d ≥ 5; otherwise NULL and excluded from the blend.

Best for: cards with high recent activity. Anchors the estimate to current market conditions when there's enough recent data to do so reliably.

7.4 Method D — Trend Projection

OLS regression of ln(price) on days_ago, fit to the last 20 sales. If the regression's R^2 is at least 0.5 (strong trend), the projection to today (days_ago = 0) is taken as a fair value candidate.

```
method_d = exp(intercept) // when days_ago = 0
```

Best for: trending markets where pure averaging methods lag reality. Examples: a hyped modern card climbing rapidly, or a once-popular card declining as supply increases.

STAGE 5

8. Adaptive Blend Weights

Default weights start at 40/40/20/0 across (EWMA, median, recent_robust, trend_projection). Three rules adjust them based on diagnostics.

Rule	Trigger	Weight Adjustment
High dispersion → lean on median	price_cov > 0.30	median +0.20, ewma –0.10, recent –0.10
Strong trend → lean on EWMA + trend	trend_r² ≥ 0.50	ewma +0.10, trend +0.20, median –0.20, recent –0.10
High recent density → lean on recent	n_last_30d ≥ 8	recent +0.20, ewma –0.10, median –0.10

Multiple rules can fire simultaneously. After applying all triggered rules, methods that returned NULL are excluded, and remaining weights are renormalized to sum to 1.0. Any weight that goes negative through rule application is clamped to zero in the renormalization step.

STAGE 6

9. Point Estimate

```
point_estimate = Σ ( weight_i × method_value_i ) for all methods with non-NULL output and weight > 0
```

The point estimate is the headline fair value — what's displayed on a card page, used in market cap calculations, compared against eBay listings for deal detection, and benchmarked in indices.

STAGE 7-8

10. Confidence Scoring

The confidence score is a 0-100 integer that quantifies how trustworthy the point estimate is. It is a weighted sum of five sub-scores, each independently 0-100.

10.1 Sub-Score 1 — Sample Adequacy

```
score_sample = 100 × ( 1 − exp( −n_total / 5 ) )
```

Saturating exponential. Half (50) at n = 3.5, near-100 at n = 15. Captures diminishing marginal value of additional samples.

10.2 Sub-Score 2 — Recency

```
if days_since_last_sale ≤ 7: score_recency = 100
else: score_recency = 100 × exp( −(days_since_last_sale − 7) × ln(2) / 30 )
```

100 within the first week, then exponential decay with 30-day halflife. A sale 30 days old scores ~50, 60 days old scores ~25, 90 days ~12.

10.3 Sub-Score 3 — Density

```
if mean_gap_days <= 14: score_density = 100
elif mean_gap_days >= 90: score_density = 0
else: linear interpolation
```

Captures whether sales cluster densely (high signal) or spread thinly (lower signal). When n_total < 2 (no gaps to compute), defaults to 50.

10.4 Sub-Score 4 — Price Stability (inverse of dispersion)

```
if price_cov <= 0.10: score_dispersion = 100
elif price_cov >= 0.50: score_dispersion = 0
else: linear interpolation
```

Tight price clustering increases confidence. Wide variance lowers it.

10.5 Sub-Score 5 — Outlier Penalty

```
score_outlier = 70 if winsorization touched any prices, else 100
```

A flat penalty: when winsorization had to clip extremes, mark down the confidence by 30%. Acknowledges that the sample contained at least one anomalous observation.

10.6 Final Confidence

```
confidence = round(
  0.25 × score_sample +
  0.30 × score_recency +
  0.15 × score_density +
  0.20 × score_dispersion +
  0.10 × score_outlier
)
```

Recency carries the highest weight at 30%. Sample size and dispersion follow at 25% and 20%. Density and outlier round out the remaining 25%. Weights are tunable via config.

10.7 Confidence Buckets

Range	Bucket	Indicative Use
80–100	very_high	Index-eligible, deal screener trustworthy
60–79	high	Index-eligible at relaxed threshold, normal display
40–59	medium	Display with caveat, exclude from indices
20–39	low	Display only with explicit warning
1–19	very_low	Background only, do not display as primary
0	none	No data; UI shows 'no recent sales'

WORKED EXAMPLES

11. The Algorithm in Practice

Example 1 — Liquid Charizard 1st Edition PSA 9

Sample: 25 sales over last 60 days, prices clustered $11.5k–$13.0k, no extreme outliers.
Winsorization: 1 sale at $13.5k clipped to p99 = $13.0k.
Diagnostics: n=25, days_since_last=2, mean_gap=2.4d, cov=0.07, trend r^2=0.15.
Rules triggered: Rule 3 (high recent density: n_30d=18 ≥ 8).
Method outputs: ewma=$12,180, median=$12,200, recent=$12,250, trend=NULL.
Final weights (after rule 3 + renormalize): ewma 0.30, median 0.30, recent 0.40.
Point estimate: $12,214.
Sub-scores: sample 99, recency 100, density 100, dispersion 100, outlier 70.
Confidence: 94 → very_high.

Example 2 — Sparse Vintage Card

Sample: 4 sales over last 18 months: $800, $850, $900, $1,100.
Winsorization: skipped (n < 5).
Diagnostics: n=4, days_since_last=180, mean_gap=135d, cov=0.13, trend NULL (n < 5).
Methods: ewma=$945, median=$875, recent=NULL, trend=NULL.
Weights (after dropping NULLs and renormalize): ewma 0.50, median 0.50.
Point estimate: $910.
Sub-scores: sample 55, recency 4, density 0, dispersion 92, outlier 100.
Confidence: 38 → low.

Example 3 — Trending Modern Card

Sample: 20 sales in last 35 days, climbing $80 → $140 (clean uptrend).
Winsorization: no clips needed.
Diagnostics: n=20, days_since_last=1, mean_gap=1.8d, cov=0.21, trend r^2=0.78.
Rules triggered: Rule 2 (strong trend) and Rule 3 (high recent density).
Method outputs: ewma=$132, median=$108, recent=$118, trend=$141.
Final weights (after rules + renormalize): ewma 0.40, median 0.10, recent 0.20, trend 0.30.
Point estimate: $130.
Sub-scores: sample 98, recency 100, density 100, dispersion 73, outlier 100.
Confidence: 94 → very_high.

Example 4 — Single Recent Sale

Sample: 1 sale yesterday at $4,200, no prior sales.
Winsorization: skipped.
Diagnostics: n=1, days_since_last=1, mean_gap=NULL, cov=NULL, trend NULL.
Methods: ewma=$4,200, median=$4,200, recent=NULL, trend=NULL.
Weights: ewma 0.50, median 0.50.
Point estimate: $4,200.

Sub-scores: sample 18, recency 100, density 50 (default), dispersion 50 (default), outlier 100.
Confidence: 56 → medium. Genuinely informative single recent sale, scored honestly.

Example 5 — Stale Sample

Sample: 3 sales 8-12 months ago, no recent activity.
Diagnostics: n=3, days_since_last=240, mean_gap=60d, cov=0.18.
Methods: ewma + median only.
Point estimate: reasonable median.
Sub-scores: sample 45, recency 0, density 30, dispersion 80, outlier 100.
Confidence: 32 → low. The value is technically computable, but downstream consumers (indices, deal screener) will exclude it based on confidence threshold.

PERSISTENCE

12. Database Schema

Every fair value computation produces one row in the `fair_values` table. The row stores the point estimate, confidence breakdown, method blend, and full diagnostics — enabling complete audit trails and downstream debugging.

Column	Type	Purpose
printing_id, grader_id, grade_id, as_of_date	FK + Date	Primary key (unique together)
value	Numeric(18,2)	Point estimate in USD
currency	Varchar(3)	Always 'USD' for now
confidence_score	SmallInt	0–100
confidence_bucket	Varchar(16)	very_low / low / medium / high / very_high / none
method_blend	JSONB	Weights used: {ewma_10: 0.4, ...}
method_outputs	JSONB	Each method's output value
n_total_sales, n_sales_last_30d, n_sales_last_90d, n_sales_last_365d	Int	Sample size diagnostics
last_sale_date, days_since_last_sale	Date / Int	Recency
mean_gap_days, price_cov	Numeric	Density and dispersion
trend_slope, trend_r_squared	Numeric	Trend diagnostics
has_outliers	Boolean	Did winsorization clip anything?
score_sample, score_recency, score_density, score_dispersion, score_outlier	SmallInt	Sub-scores 0–100
created_at, updated_at	Timestamp	Audit

OPERATIONS

13. Daily Rolling Computation

Fair values are computed daily for every (printing, grader, grade) tuple that has any sales history. The pipeline is idempotent — re-running for the same as-of-date produces identical results and upserts cleanly via the unique key.

Daily flow:

- Cron triggers `scripts/calculate_fair_values.py` overnight after raw price ingestion completes.
- For as_of_date = today: iterate all (printing, grader, grade) tuples with at least one historical sale.
- For each tuple, run the algorithm and upsert one row into `fair_values`.

- Job report (success count, failure count, duration) written to `job_runs` table.

Backfill is handled via the same script with a `--start-date` / `--end-date` range argument. Backfill scope (e.g., last 30 days, last 365 days, or full history) is a deployment decision separate from this ticket.

INTEGRATION

14. Downstream Consumers

Consumer	How It Uses Fair Value
Market Cap Engine	Multiplies fair_value × population for each (printing, grader, grade) atom
Indices	Filter by confidence ≥ 60 for inclusion; weight by graded market cap
Deal Screener	Compare eBay listing prices against fair_value to detect undervalued listings
Grading ROI Calculator	Reads fair_value at each grade level for a printing
Card Pages (Frontend)	Display point estimate + confidence badge + method breakdown on hover
AI Commentary	Top movers determined by changes in fair_value over time windows

LIMITATIONS

15. Known Limitations

- FX rates hardcoded for now. Real-time ECB ingestion is a separate future ticket. The hardcoded rates may drift from market over time.
- Sale source not weighted. A TCGPlayer Direct sale and an eBay private sale are treated equally. Source-aware weighting could be added later.
- No cross-grade signal borrowing. PSA 8 trends do not inform PSA 9 confidence even when PSA 9 has thin data. Bayesian shrinkage toward category prior is a future enhancement.
- Sale type (auction vs BIN) not used. `raw_prices.sale_type` exists but is not currently filtered. May add later.
- Trend regression assumes log-linear drift. Doesn't model regime changes (e.g., before/after a set release).
- Confidence sub-score weights are fixed defaults. Optimal weights could be calibrated empirically against actual price prediction error.

This methodology document is the spec for ticket SCRUM-XX in the TCGBerg backlog. Implementation details (table DDL, module structure, test coverage) are captured in the ticket itself.

Companion documents: Index Methodology PDF, Pokemon Data Sources PDF.
Document version 1.0 — May 2026.