Comprehensive MBS Pool Analysis - Business Formula Documentation

For Asset-Backed Securities Collateralized by Mortgage Pools

Executive Summary

This documentation provides detailed business formulas and methodologies for analyzing mortgage-backed securities (MBS) backed by pools of residential mortgages. The analysis framework covers pool characteristics, cash flow modeling, prepayment analysis, and risk assessment metrics essential for MBS trading and portfolio management.

---

1. Pool Structure Analysis

1.1 Weighted Average Coupon (WAC)

The WAC represents the weighted average interest rate of all mortgages in the pool, weighted by outstanding principal balance.

Formula:

$$WAC = \Sigma(\text{Mortgage Coupon Rate} \times \text{Outstanding Balance}) / \Sigma(\text{Outstanding Balance})$$

Implementation in Notebook:

```python
# From instrument_data structure
WAC = Σ(Coupon × CurrentUPB) / Σ(CurrentUPB)
```

Business Context:

- Critical for determining pass-through rate to investors
- Affects interest rate sensitivity and prepayment behavior
- Higher WAC pools typically exhibit higher prepayment sensitivity

1.2 Weighted Average Maturity (WAM)

Measures the average remaining term to maturity of mortgages in the pool.

Formula:

```
WAM = Σ(Remaining Months to Maturity × Outstanding Balance) / Σ(Outstanding Balance)
```

Current Implementation:

```python
# Direct from data - WAM field represents weighted average months
WAM_Years = WAM / 12  # Convert to years for duration calculations
```

1.3 Weighted Average Loan Age (WALA)

Measures how long the mortgages have been outstanding since origination.

Formula:

```
WALA = Σ(Loan Age in Months × Outstanding Balance) / Σ(Outstanding Balance)
```

Seasoning Impact on Prepayments:

- **0-12 months**: Low prepayment (borrowers settling in)
- **12-30 months**: Increasing prepayment (refinancing eligibility)
- **30-60 months**: Peak prepayment period
- **60+ months**: Declining prepayment (seasoned borrowers)

---

2. Prepayment Analysis Framework

2.1 Conditional Prepayment Rate (CPR)

CPR measures the annual rate at which mortgage principal is prepaid.

Formula:

```
CPR = 1 - (1 - SMM)^12

Where SMM (Single Monthly Mortality) = Monthly Prepayment Rate
```

Multi-Period CPR Analysis (Notebook Implementation):

```python
```

```
def calculate_prepayment_risk_metrics(instrument):
    cpr_values = [
        instrument.get('CPR1Month'),
        instrument.get('CPR3Month'),
        instrument.get('CPR6Month'),
        instrument.get('CPR12Month'),
        instrument.get('CPR24Month')
    ]

    # CPR Volatility - measure of prepayment uncertainty
    cpr_volatility = np.std(cpr_values)

    # CPR Trend - directional movement in prepayment speeds
    time_periods = [1, 3, 6, 12, 24]
    slope, _, _, _, _ = linregress(time_periods, cpr_values)
    cpr_trend = slope
```

2.2 Prepayment Speed Assessment

Speed Score Calculation:

```
Speed Score = (Current CPR / Historical Average CPR) × 100

Interpretation:
- Score > 120: Fast prepayment environment
- Score 80-120: Normal prepayment range
- Score < 80: Slow prepayment environment
```

Geographic Risk Factors:

```
python

state_risk_factors = {
    'CA': 1.2,  # High refinancing activity, rate-sensitive borrowers
    'NY': 1.3,  # Highest refinancing activity, sophisticated borrowers
    'FL': 1.1,  # Moderate-high, vacation home refinancing
    'TX': 1.0,  # Baseline, diverse economy
    'WA': 1.0,  # Tech sector influence, moderate activity
    'AZ': 0.9   # Lower refinancing rates, price-sensitive market
}
```

2.3 Seasoning Risk Model

Seasoning Curve Implementation:

```python
def calculate_seasoning_risk(wala_months):
    if 24 <= wala_months <= 60:
        return 1.2  # Peak refinancing period
    elif 12 <= wala_months < 24:
        return 1.1  # Ramp-up period
    else:
        return 0.9  # Either too new or too seasoned
```

Business Rationale:

- **Months 0-12**: Borrowers adjusting to new home, limited refinancing
- **Months 12-24**: Qualification for no-cash-out refinancing
- **Months 24-60**: Peak period for rate/term refinancing
- **Months 60+**: Remaining borrowers less rate-sensitive

2.4 Combined Prepayment Risk Score

Formula:

```
Combined Risk Score = Speed Score × Geographic Risk × Seasoning Risk × Pool Characteristics

Where Pool Characteristics include:
- Loan size factor
- FICO score distribution
- Loan-to-value ratios
- Property type concentration
```

---

3. Cash Flow Modeling

3.1 Accrued Interest Calculation

Standard 30/360 Day Count:

```python
```

```
def calculate_accrued_interest(issue_date, settlement_date, coupon_rate, frequency=12):
    # 30/360 Convention Rules:
    # - Each month = 30 days
    # - Each year = 360 days
    # - Adjust end-of-month dates per convention

    days_accrued = days_30_360(last_coupon_date, settlement_date)
    days_in_period = 360 // frequency

    accrued_interest = (coupon_rate / frequency) * (days_accrued / days_in_period)
    return max(0, accrued_interest)
```

Business Application:

- Used for trade settlement calculations
- Essential for accurate P&L attribution
- Affects total return calculations

3.2 Dirty Price Calculation

Formula:

```
Dirty Price = Clean Price + Accrued Interest

Where:
- Clean Price = Quoted market price (excludes accrued interest)
- Accrued Interest = Interest earned from last payment date to settlement
```

Implementation:

```
python

def calculate_dirty_price(clean_price, accrued_interest):
    return clean_price + accrued_interest
```

---

4. Interest Rate Risk Metrics

4.1 Modified Duration (Simplified Model)

Formula:

Modified Duration ≈ WAM (years) / (1 + Coupon Rate / 200)

Limitations of Simplified Model:

- Does not account for embedded prepayment options
- Ignores convexity effects from prepayment behavior
- Assumes parallel yield curve shifts

Enhanced Duration Model (Recommended):

Effective Duration = (Price at -50bp - Price at +50bp) / (2 × Base Price × 0.005)

Where prices account for prepayment option value

4.2 Convexity Approximation

Current Implementation:

Convexity ≈ Duration^2 / (1 + Coupon Rate / 100)

Note: MBS typically exhibit negative convexity due to embedded prepayment options. The approximation may not capture this critical characteristic.

4.3 Option-Adjusted Spread (OAS)

Simplified Implementation:

```python
def calculate_option_adjusted_spread(instrument, market_yield=4.5):
    coupon = instrument.get('Coupon', 3.5)
    base_spread = coupon - market_yield

    # Prepayment risk adjustment
    prepay_metrics = calculate_prepayment_risk_metrics(instrument)
    prepay_adjustment = (prepay_metrics['Combined_Risk_Score'] / 100 - 1) * 50

    oas = base_spread * 100 - prepay_adjustment  # Convert to basis points
    return oas
```

Production OAS Model Requirements:

1. Monte Carlo interest rate simulation (1,000+ paths)
2. Prepayment model calibrated to current market conditions
3. Volatility surface for interest rate modeling
4. Credit spread assumptions

---

5. Risk Assessment Framework

5.1 Pool Quality Metrics

Loan Count Concentration:

```python
# From notebook data structure
concentration_risk = 1 / sqrt(LoanCount)

# Higher loan count = better diversification = lower concentration risk
```

Geographic Concentration:

```python
# Primary state concentration (from StatePercentage field)
geographic_concentration = max(StatePercentage) / 100

# >50% concentration in single state increases risk
```

5.2 Credit Risk Indicators

Pool Seasoning Analysis:

```python
```

```
def assess_pool_maturity(wala, maturity_term):
    remaining_life = maturity_term - wala
    seasoning_ratio = wala / maturity_term

    if seasoning_ratio > 0.7:
        return "Highly Seasoned"  # Lower credit risk, lower prepayment risk
    elif seasoning_ratio > 0.3:
        return "Moderately Seasoned"
    else:
        return "Newly Originated"  # Higher uncertainty
```

---

6. Data Assumptions and Limitations

6.1 Critical Assumptions

Interest Rate Environment:

- Market yield assumption: 4.5% (10-year Treasury proxy)
- Parallel yield curve shifts for duration calculations
- No consideration of yield curve steepening/flattening

Prepayment Model Assumptions:

- Historical CPR patterns predict future behavior
- No modeling of economic cycle impacts
- Geographic factors remain constant
- No consideration of loan-level characteristics (FICO, LTV, DTI)

Pool Composition Assumptions:

- Uniform distribution of loan characteristics within pools
- No consideration of loan purpose (purchase vs. refinance)
- Property type assumed to be primarily single-family residential

6.2 Model Limitations

Duration/Convexity:

Current Model: Basic approximation
Missing Elements:
- Prepayment option value
- Path-dependent cash flows
- Credit risk impact
- Liquidity premium adjustments

OAS Calculation:

Current Model: Static spread over Treasury
Missing Elements:
- Monte Carlo simulation (1,000+ interest rate paths)
- Prepayment model calibration
- Credit loss expectations
- Liquidity adjustments

Prepayment Analysis:

Current Model: Historical pattern analysis
Missing Elements:
- Current rate environment impact
- Borrower incentive calculations (rate differential)
- Housing market conditions
- Loan-level modeling (burnout, SATO effects)

---

7. Trading Applications

7.1 Relative Value Analysis

Cross-Pool Comparison:

python

```python
def compare_pools(pool1, pool2):
    metrics = {
        'OAS_Difference': pool1['OAS'] - pool2['OAS'],
        'Duration_Match': abs(pool1['ModifiedDuration'] - pool2['ModifiedDuration']),
        'Prepay_Risk_Delta': pool1['PrepaymentRiskScore'] - pool2['PrepaymentRiskScore']
    }
    return metrics
```

7.2 Portfolio Construction Guidelines

Duration Matching:

```
Target Portfolio Duration = Σ(Pool Duration × Pool Weight)
Duration Contribution = Pool Duration × Pool Market Value / Total Portfolio Value
```

Risk Budgeting:

```
Risk Budget Allocation:
- Interest Rate Risk: 60-70% of total risk
- Prepayment Risk: 20-30% of total risk
- Credit Risk: 5-10% of total risk
- Liquidity Risk: 5-10% of total risk
```

7.3 Hedging Considerations

Interest Rate Hedging:

```python
# Treasury hedge ratio calculation
treasury_hedge_ratio = mbs_duration / treasury_duration * mbs_market_value / treasury_market_value

# Account for basis risk between MBS and Treasury movements
basis_adjustment = historical_correlation * volatility_ratio
adjusted_hedge_ratio = treasury_hedge_ratio * basis_adjustment
```

---

8. Operational Guidelines

8.1 Data Quality Checks

Required Validations:

1. WAM + WALA ≤ Original Term
2. Current UPB ≤ Original UPB
3. CPR values within reasonable ranges (0-50%)
4. Geographic percentages sum to ≤100%

Missing Data Handling:

```python
# Default values for missing data
DEFAULT_VALUES = {
    'market_yield': 4.5,
    'clean_price': 100.0,
    'cpr': 5.0,
    'state_risk': 1.0
}
```

8.2 Model Validation

Backtesting Framework:

- Compare predicted vs. actual prepayment speeds
- Validate duration estimates against market price movements
- Assess OAS model accuracy vs. dealer quotes

Stress Testing:

- Interest rate scenarios: +/-200bp parallel shifts
- Prepayment scenarios: 0.5x to 2.0x base case speeds
- Credit scenarios: 1-3x historical loss rates

---

9. Recommendations for Enhancement

9.1 Immediate Improvements

1. **Integrate loan-level data** for accurate pool modeling
2. **Implement proper OAS model** with Monte Carlo simulation
3. **Add credit risk assessment** with loss forecasting
4. **Include liquidity metrics** for trading optimization

9.2 Advanced Analytics

1. **Machine learning prepayment models** using borrower characteristics
2. **Real-time market data integration** for dynamic pricing
3. **Scenario analysis tools** for stress testing
4. **Attribution analysis** for performance measurement

Conclusion

This documentation provides a comprehensive framework for MBS pool analysis, highlighting both the current implementation's capabilities and its limitations. Traders should use these metrics as screening tools while understanding the simplified nature of certain calculations. For production trading decisions, consider implementing the enhanced methodologies outlined in the recommendations section.

Key Takeaways:

- Pool characteristics drive prepayment behavior and risk
- Simplified models provide directional guidance but have limitations
- Geographic and seasoning factors significantly impact performance
- Regular model validation and enhancement are essential for accuracy

---

Document Version: 1.0
Last Updated: August 2025
Next Review: Quarterly