R Analytics Module 1: Shock-Proof Rate Benchmark

1. Technical Significance

The **Shock-Proof Rate Benchmark** is designed to separate the "signal" (true market rate) from the "noise" (temporary shocks due to fuel spikes, strikes, or holidays). Standard averaging methods fail because they treat all variance as equal. Our approach uses **Bayesian Structural Time Series (BSTS)** combined with **GARCH (Generalized Autoregressive Conditional Heteroskedasticity)** to mathematically distinguish between the stable base rate and the volatility premium.

2. The R Technique: BSTS + GARCH

We employ a dual-model approach in R:

A. Bayesian Structural Time Series (BSTS)

- **Purpose:** To estimate the "Base Level" (the stable, underlying rate).
- **Algorithm:** `bsts` package in R.
- **Mechanism:** It deconstructs the time series into:
  - *Local Linear Trend:* The direction the rate is moving naturally.
  - *Seasonality:* Recurring patterns (e.g., rates rising in Q4).
- **Why it's superior:** Unlike ARIMA, BSTS handles changing trends dynamically and provides a distribution of probable outcomes, not just a single point.

B. GARCH(1,1) Volatility Modeling

- **Purpose:** To calculate the "Shock Component" (the risk premium).
- **Algorithm:** `rugarch` package in R.
- **Mechanism:** It models the *variance* of the error terms from the BSTS model. It assumes that volatility clusters (periods of high risk follow other periods of high risk).
- **Result:** A dynamic "Shock Premium" (e.g., + 150) that represents the current market instability.

3. Workflow & Architecture

1. **Data Ingestion:**
   - The user selects a lane (e.g., "Mumbai-Delhi").
   - The frontend sends a request to `backend/r_analytics_service.py`.
2. **Data Preparation:**
   - The system retrieves 24 months of historical contract rates and spot market rates from the database.

3. **R Execution (`benchmarking.R`):**
    - The Python backend spawns an R subprocess.
    - It passes the historical vectors as JSON to R.
    - R runs the `benchmark_analysis()` function.
4. **Statistical Processing:**
    - BSTS fits the trend.
    - GARCH analyzes the residuals for volatility.
    - R computes the lower/upper 95% confidence intervals.
5. **Output:**
    - R returns a JSON object with: `forecast_base`, `volatility_shock`, and `confidence_interval`.

4. Sample Data & Results

Input Data (Sample)

```
{
  "lane": "Mumbai-Delhi",
  "history_months": 24,
  "contract_rates": [3000, 3020, 3050, 3040, 3100, ...],
  "market_rates": [2800, 2850, 2900, 3200, 3150, ...]
}
```

R Processing Output

```
{
  "forecast": [3250],         // The predicted stable Base Rate
  "volatility": [150],        // The calculated Shock Premium
  "total_benchmark": 3400,    // Base + Shock
  "is_disruption": true,      // GARCH volatility > Threshold
  "confidence": 96.5          // Probability score
}
```

User Facing Result

- **Base Rate:** 3,250 (What you *should* pay in a normal market)
- **Shock Premium:** + 150 (Extra cost due to current volatility)
- **Total Benchmark:** 3,400
- **Verdict:** “Contracted” (The market is currently unstable, proceed with caution)