# Curating Riot's TFT Match Data: A Reproducible Data Workflow Project

Joaquin Ugarte

## Overview

The objective of this project is to develop a comprehensive, reproducible workflow for curating match data from Riot Games' Teamfight Tactics (TFT) public API. This project addresses the critical need for structured, analysis-ready datasets in gaming analytics while serving as an ideal application of course concepts in data curation.

The project's use case centers on creating a robust data infrastructure that enables researchers, data scientists, and gaming analysts to conduct meaningful analyses of player behavior, game evolution, and competitive dynamics in TFT. The research questions this dataset will support include: How do champion and item choices evolve over time? What factors contribute to successful team compositions? How do player strategies adapt to game updates and balance changes?

By developing an automated, quality-assured data pipeline, this project will provide the gaming research community with a valuable resource while demonstrating best practices in data curation, reproducibility, and API-based data collection workflows.

## Plan: Data Lifecycle Integration

This project follows a comprehensive data lifecycle model encompassing the collection, processing, analysis preparation, preservation, and sharing, incorporating key data curation concepts including data abstractions, metadata standards, provenance documentation, and reproducible workflows.

### Phase 1: Planning and Infrastructure Design (Week 1)

During this intensive initial phase, I will establish the project architecture and develop the foundational data collection infrastructure simultaneously. This includes analyzing Riot's API documentation, understanding TFT data structures through relational and hierarchical abstractions and implementing basic API interaction scripts with rate limiting to comply with Riot API limitations. I will create specifications for data collection parameters, quality metrics, and output formats and schemas, while ensuring full compliance with Riot's Developer Terms of Service.

### Phase 2: Data Collection System and Quality Framework (Weeks 2-4)

This phase integrates automated data collection with comprehensive quality assessment tools, applying **relational and tree-based abstractions** learned in the course. The system will implement **data integration and cleaning** methodologies with focus on the LA2 (Latin America South) region primarily, with NA1 (North America) integration planned if API quotas and time of extraction allows it. The workflow will include automated detection of formatting changes, validation using tree-structured data abstractions to handle hierarchical match data, identification of outliers, and handling of missing fields. An automated scheduling system using cron jobs will be implemented for weekly data updates, with alternative scheduling solutions including Apache Airflow.

### Phase 3: Data Organization and Metadata Implementation (Weeks 5-6)

This phase applies **metadata standards** and **identity/identifier** concepts from the course, transforming raw API responses into analysis-ready formats using relational data abstractions. This includes standardizing variable names, creating normalized data structures, implementing efficient storage formats (Parquet, Jsonl or Csv), and developing comprehensive metadata documentation following schema.org standards. Identity and identifier systems will be established for match tracking and data provenance.

### Phase 4: Workflow Integration and Preservation Strategy (Weeks 7-8)

Using Snakemake, I will implement the complete workflow as a reproducible pipeline with full provenance tracking. This includes parameterization for different collection periods, automated testing procedures, validation against known datasets, and implementation of data preservation strategies ensuring long-term accessibility and integrity.

### Phase 5: Continuous Updates and Documentation (Weeks 9-10)

The system will be deployed with weekly automated updates via cron scheduling, ensuring continuous data population. Comprehensive documentation will be prepared, including user guides, data dictionaries, and workflow documentation that emphasizes reproducibility and provenance.

### Phase 6: Testing, Validation and Publication (Weeks 11-12)

Final testing of the complete system including the automated update mechanism, validation of data quality over multiple collection cycles, and publishing the complete workflow and dataset via GitHub with appropriate licensing, citation guidelines, and usage documentation.

## Data Sources

The primary data source is Riot Games' TFT API (https://developer.riotgames.com/), which provides comprehensive match data structured in hierarchical formats ideal for tree-based data abstractions. The project will focus primarily on the LA2 (Latin America South) region, with potential expansion to NA1 (North America) pending API quota availability.

**Core Data Structures Include:**

- **Match Details**: Game duration, dates, participant information, placement outcomes
- **Player Data**: Summoner information, rank, regional data structured in relational format
- **Composition Data**: Champion selections, item builds, synergy combinations
- **Timeline Data**: Round-by-round progression represented as temporal trees

**API Endpoints:**

- `/tft/match/v1/matches/{matchId}` for detailed hierarchical match information
- `/tft/league/v1/entries/{tier}/{division}` for ranked player data
- `/tft/summoner/v1/summoners/{encryptedSummonerId}` for player identification and linking

**Automated Update System:**
Weekly data updates will be implemented using automated scheduling with multiple implementation options:

1. **Primary**: Cron jobs on dedicated Linux machine with error handling and logging
2. **Alternative**: Apache Airflow or alternative scheduler

Secondary data sources include Riot's Data Dragon service for static game data validation and community datasets for quality assessment benchmarking.

## Team Structure

This is an individual project where I will assume full responsibility for all components:

**Primary Responsibilities:**

- API integration and data collection system development
- Quality assessment framework design and implementation
- Snakemake workflow development and testing
- Documentation creation and maintenance
- GitHub repository management and publication

**Skills Alignment:**

- Programming: Python for API interaction, data processing, and workflow development
- Data Management: Experience with data validation, cleaning, and structured storage
- Documentation: Technical writing and metadata standards implementation

- Version Control: Git/GitHub for project management and publication

## Timeline

### Project Schedule Overview

| Phase | Duration | Focus Area | Key Deliverables |
|---|---|---|---|
| **Planning & Infrastructure** | Week 1 | Foundation | API integration, compliance framework |
| **Collection & Quality** | Weeks 2-4 | Data Pipeline | Automated collection, quality validation |
| **Organization & Metadata** | Weeks 5-6 | Data Structure | Relational models, metadata standards |
| **Workflow & Preservation** | Weeks 7-8 | Reproducibility | Snakemake pipeline, provenance tracking |
| **Automation & Documentation** | Weeks 9-10 | Operations | Continuous updates, comprehensive docs |
| **Validation & Publication** | Weeks 11-12 | Finalization | Testing, GitHub publication |

### Detailed Weekly Breakdown

```
Week 1: Foundation Setup
├── Riot API documentation analysis and compliance review
├── Basic Python infrastructure for API interaction
├── Rate limiting and error handling framework
└── Initial data schema design

Weeks 2-4: Core Development Sprint
├── Week 2: LA2 region data collection implementation
│   ├── Match data retrieval system
│   ├── Player identification and linking
│   └── First automated data collection run
├── Week 3: Quality assessment framework
│   ├── Tree-based data validation
│   ├── Missing field detection algorithms
│   └── Anomaly identification systems
└── Week 4: Integration and testing
    ├── End-to-end pipeline testing
    ├── Data quality metrics establishment
    └── NA1 region expansion assessment
```

Weeks 5-6: Data Architecture
├── Week 5: Relational data modeling
│   ├── Normalized database schema design
│   ├── Efficient storage format implementation (Parquet)
│   └── Data transformation pipelines
└── Week 6: Metadata and standards
    ├── Schema.org metadata implementation
    ├── Data dictionary development
    └── Identity and identifier systems

Weeks 7-8: Workflow Engineering
├── Week 7: Snakemake pipeline development
│   ├── Reproducible workflow implementation
│   ├── Parameterization for flexibility
│   └── Provenance tracking integration
└── Week 8: Preservation strategy
    ├── Data backup and versioning systems
    ├── Long-term accessibility planning
    └── Workflow validation and testing

Weeks 9-10: Operational Deployment
├── Week 9: Automation implementation
│   ├── Cron job scheduling setup
│   ├── Weekly update system deployment
│   ├── Automated quality reporting
│   └── Error notification systems
└── Week 10: Documentation and guides
    ├── Comprehensive user documentation
    ├── API usage guidelines
    ├── Troubleshooting guides
    └── Data citation standards

Weeks 11-12: Validation and Release
├── Week 11: System validation
│   ├── Multi-week automated collection testing
│   ├── Data quality validation across cycles
│   ├── Performance optimization
│   └── Security and compliance review
└── Week 12: Publication and finalization

```
├── GitHub repository preparation
├── Final documentation review
└── Dataset publication
```

### Automated Update Schedule

**Weekly Data Collection Cycle:**

- **Trigger**: Every Sunday at 2:00 AM (minimize API load)
- **Duration**: **To be defined** depending on match volume
- **Expected Volume**: **To be defined**
- **Quality Check**: Automated validation within 1 hour of collection
- **Reporting**: Weekly summary with data quality metrics in a dashboard visualization tool or automated email reports

### Key Milestones & Deadlines

| Week | Milestone | Success Criteria |
|---|---|---|
| **2** | First Data Collection | Successfully collect 100+ matches from LA2 |
| **4** | Quality Framework Complete | Automated validation detecting 95%+ data issues |
| **6** | Metadata Standards | Full schema.org compliance and documentation |
| **8** | Reproducible Workflow | Complete Snakemake pipeline with provenance |
| **9** | Automation Live | Weekly updates operational with monitoring |
| **12** | Project Publication | GitHub release with comprehensive dataset |

### Risk Mitigation Timeline

- **Week 1**: API key approval contingency (backup manual collection)
- **Week 3**: NA1 integration decision point based on quota usage
- **Week 5**: Data schema evolution challenges if TFT game updates change API structure
- **Week 6**: Metadata standard adjustment if schema.org proves insufficient
- **Week 9**: Fallback scheduling options if cron implementation fails
- **Week 10**: Documentation completeness review checkpoint
- **Week 11**: Manual validation if automated testing reveals issues

## Constraints

Several constraints must be carefully managed throughout this project:

**Technical Constraints:**

- Riot API rate limits (100 requests per 2 minutes, 20,000 per 24 hours for production keys)
- Data volume management and storage considerations
- Dependency on Riot's API stability and availability

**Legal and Ethical Constraints:**

- Strict adherence to Riot's Terms of Service and API usage policies
- Player privacy considerations and data anonymization requirements
- Commercial use restrictions that may limit dataset distribution

**Resource Constraints:**

- Limited to personal API key quotas initially
- Computational resources for large-scale data processing
- Storage requirements for comprehensive historical data

## Gaps and Information Needs

Several areas require additional investigation and input:

**Technical Gaps:**

- Optimal sampling strategies for representative data collection across regions and skill levels
- Specific quality metrics most relevant to TFT match analysis
- Performance optimization for large-scale data processing workflows

**Domain Knowledge Gaps:**

- Deep understanding of TFT meta-game evolution patterns for quality assessment
- Community research priorities to ensure dataset utility
- Best practices for gaming data anonymization and privacy protection

**Implementation Gaps:**

- Optimal tree-based representation strategies for hierarchical TFT match data
- Integration of course concepts including ontological frameworks for TFT game states
- Best practices for provenance tracking in automated data collection workflows
- Schema evolution strategies for handling API changes over time
- Appropriate metadata standards balancing discoverability with gaming data specificity

To address these gaps, I will engage with the TFT research community, consult Snakemake documentation and tutorials, and potentially collaborate with Riot's developer relations team for guidance on best practices and compliance requirements.

---

## References


Riot Games. (2024). *Riot Developer Portal*. https://developer.riotgames.com/

Riot Games. (2024). *Teamfight Tactics API Documentation*. https://developer.riotgames.com/apis#tft-match-v1

Schema.org Community Group. (2024). *Schema.org*. https://schema.org/

Snakemake 9.11.2 Documentation. https://snakemake.readthedocs.io/en/stable/

Köster, J., & Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19), 2520-2522. https://doi.org/10.1093/bioinformatics/bts480