

Final Project Report

Title : Curating and Integrating Solo Queue and International Tournament Data in League of Legends Online Game

Student : Juhwan Song

Date: December 10, 2025

1. Introduction & Project Overview

League of Legends (LoL) is a global game released in 2009, with a professional esports scene starting in 2011. Over more than ten years of updates, the game has become more complex, and the gap between professional players and general users has grown. Skilled players handle deeper mechanics and strategy, while new or less experienced users face higher barriers to learning. This creates concerns for long-term engagement, especially for users who want to understand high-level play but lack simple and clear resources.

This project aims to reduce this gap by creating a curated, data-driven link between Solo Queue (SoloQ) gameplay and professional matches. By identifying patterns such as laning efficiency, macro movement, objective timing, and common playstyles, the curated dataset helps general users see how professionals approach the same situations in the game. The goal is to provide clear insights that make high-level gameplay easier to understand.

A major challenge is that SoloQ and professional data come from different providers and use different schemas. They also differ in structure, granularity, and metadata. Because of this, they cannot be compared directly without careful data curation to fix these differences and track provenance.

To address this, the project builds an end-to-end pipeline that acquires, cleans, transforms, and merges both datasets into one unified format. The curated results are shown through a visualization platform that lets users explore the differences between SoloQ and professional play in an intuitive way. Through this workflow and visualization, the project supports user learning and improves access to advanced gameplay concepts in [Github repository](#).

2. Dataset Profile

This project integrates two distinct data sources: Solo Queue (SoloQ) gameplay data collected from Riot Games' public API and professional match data sourced from Oracle's Elixir. Although both datasets originate from the same game environment, they differ in acquisition methods, schema design, granularity, and metadata conventions. Understanding these differences is essential for building a unified dataset that enables meaningful comparisons between professional players and everyday users.

2.1 Solo Queue Dataset

Solo Queue data in this project is obtained directly from the Riot Games Developer API and is delivered in structured JSON format across multiple endpoints. Each endpoint provides a different level of granularity and is used together to reconstruct a complete representation of player identity, ranked status, and match-level behavior.

2.1.1 Player Account Information

Player-level identifiers are retrieved through the Account API. This endpoint returns a persistent global identifier (**puuid**), which is the primary key used across all subsequent data requests. Additional metadata such as in-game name and region tag are included only as contextual attributes. The minimal example below illustrates the structure.

Field	Type	Description
puuid	string	Global, persistent player identifier used across all Riot endpoints.
gameName	string	Player's in-game name.
tagLine	string	Region routing tag(e.g., KR1, NA1). Used for routing API requests.

The **puuid** is stable across patches and seasons and serves as the core identifier for linking a user to all SoloQ match records.

2.1.2 Summoner(Player) Profile Information

The Summoner API returns profile-level attributes such as summoner level, profile icon, and revision timestamp. This data is used primarily for player metadata enrichment and consistency checks.

Field	Type	Description
puuid	string	Links profile metadata to the player's global identifier.
profileIconId	Integer	ID of the profile icon selected by the player.
RevisionDate	Long(timestamp)	Last account update timestamp(ms).

Although not directly used in gameplay analysis, this endpoint ensures identifier consistency and validates that account records are correctly aligned before match data acquisition.

2.1.3 Ranked Information

Ranked status is retrieved through the League API, which provides competitive tier, division, league points, and win-loss records. This information is essential for contextualizing SoloQ match performance and filtering players by skill segment. Example schema:

Field	Type	Description
leagueId	String	Unique league grouping identifier
QueueType	String	Game category
Tier	String	Competitive tier(Iron → Challenger)
Rank	String	Division within tier (I, II, III, IV).
Puuid	String	Player identity link to account_info.
veteran/hotStreak/etc.	Boolean	Status flags indicating player consistency patterns.

This endpoint highlights high-level behavioral differences between player tiers and supports later comparative analysis between SoloQ users and professional players.

2.1.4 Match and Timeline Data

Match data (via **match-v5**) contains complete metadata for each game, including champion selections, team compositions, performance statistics, and game-level outcomes. Timeline data provides frame-by-frame state transitions such as gold totals, XP progression, skill order, item purchases, and map movement events.

Field	Type	Description
Metadata.matchID	String	Global unique match identifier (region + numerical ID).
Metadata.participants	List[string]	List of 10 participants puuids.
Info.gameCreation	Long	Match creation timestamp.
Info.gameDuration	Integer	Total match time in seconds.
Info.gameVersion	String	Patch version (e.g., 14.1). Used for patch-aligned analysis.
Info.participants[]	Object	Contains >400 player-level variables including kills, gold.
Info.teams[]	Object	Team-level stats including objectives taken.
Info.frames[]	List	Per-minute game state snapshots.
Info.events[]	List	Kill, assist, objective, item purchase, and position events.

This endpoint is the foundation for all gameplay analysis.

2.1.5 Acquisition Method

SoloQ data is collected through a fully automated pipeline built with **GitHub Actions**, which periodically retrieves new match IDs by querying a player's puuid, then fetches the corresponding match and timeline records. The workflow includes:

- Controlled request scheduling to comply with Riot API rate limits
- Automatic retry logic for 429 throttling or temporary downtime
- Logging of failed requests for reprocessing
- Versioning of all downloaded raw JSON files for provenance tracking

This ensures reproducible, large-scale data acquisition while maintaining compliance with public API constraints.

2.2 Professional Dataset

The professional dataset includes match records from the 2025 Spring, Summer, and Worlds seasons. The data comes in CSV format from Oracle's Elixir. Each row shows one player's performance in a game and includes basic metadata such as league, split, date, match ID, team, role, and champion.

The dataset also contains detailed competitive statistics, including kills, deaths, assists, damage, gold, vision, objectives, and early-game indicators. Unlike SoloQ data, which uses nested JSON and timeline frames, the professional data is already flat and uses consistent column names across major leagues. These stable metrics make it easy to compare champion choices, lane outcomes, and decision-making patterns between professional players and SoloQ users.

Field Group	Example Fields	Description
Match Metadata	Gameid, league, year, split, patch, date	Identifies the competitive environment and match context.



Participant Info	Playername, playerid, teamname, position	Describes player identity, team affiliation.
Champion & Draft	Champion, ban1-ban5	Champion selection and drafting phase.
Core Combat Stats	Kills, deaths, assists, teamkills	Standard indicators of in-game performance
Damage & Gold Economy	Damageto champions, damageshare, totalgold	Measures efficiency in dealing damage and converting resources.
Vision Control	Vision scorewardsplaced	Quantifies map control and team utility contributions.
Objective Control	Dragons, towers, barons	Captures macro-level decisions and coordinated team play.
Early -Game Differentials	Goldat10	Lane outcomes and early-game tempo indicators used for comparative analysis.

2.3 Unified Schema

To compare SoloQ and professional data fairly, both sources were aligned into one shared schema. SoloQ JSON files and professional CSV files were matched using only fields that exist in both datasets. All personal identifiers, such as puuid and player names, were removed. Different formats were normalized to the same structure.

The unified schema focuses on gameplay fields that both datasets share, such as combat efficiency, resource use, vision, objective control, and early-game lane outcomes. These common fields allow clear and fair comparisons across groups. Full variable definitions, including types, units, and missing-value rules, are listed in Appendix D (Data Dictionary).

Table 4. Unified Schema Overview

Unified Field Group	Example Metrics	Description
Role & Context	Role, side, champion	Role labels and champion identity
Combat Efficiency	Kills, deaths, assits, kda	Combat performance.
Damage Metrics	Damage_to_champions, damage_share	Measures of offensive output
Economy Metrics	Total_gold, Creep_score	Resource generation and farming efficiency
Vision Control	Vision_score, wards_placed	Map control and utility contribution
Objective Impact	Dragons, barons, towers	Contribution to macro-objective control
Early-Game Outcomes	Glod_diff_at_10_minutes	Lane-phase performance differentials.
Dataset Source	Dataset_type(soloq/pro)	Indiciates which dataset the row originates from.

3. Data Modeling & Schema Harmonization

3.1 Raw Data Structures: JSON (SoloQ) vs. CSV (Professional)

SoloQ data contains **minute-level timeline frames** provided as nested JSON. Each match includes a list of 10 participants, and for every minute, SoloQ offers detailed frame data such as gold, experience, and position:

- `info.participants[10]`
- `info.frames[0..n]` (1-minute intervals)
- per-minute gold / XP / position / event logs

In contrast, professional data does not include a timeline. It is delivered as flat CSV rows, and minute-by-minute progression is not available. Instead, professional data provides **pre-defined checkpoint statistics** at major timestamps (e.g., 10, 15, 20, 25 minutes):

- `goldat10, xpdiffat15, csdiffat20`
- already aggregated metrics like `dpm, vspm, damageshare`

Because SoloQ offers full 1-minute frames while professional data only provides coarse time snapshots, the two datasets cannot be directly aligned. This structural mismatch requires harmonization, where only comparable gameplay metrics are extracted and normalized into a unified schema.

3.2 Field Normalization & Naming Alignment

To make SoloQ JSON and professional CSV comparable, overlapping gameplay variables were mapped into a shared naming scheme. The table below summarizes representative examples of how heterogeneous fields from both sources were unified.

Unified Field Name	SoloQ Field(JSON)	Professional Field(CSV)
<code>player_damage</code>	<code>totalDamageDealToChampions</code>	<code>DamagetoChampions</code>
<code>role</code>	<code>teamPosition</code>	<code>Role</code>
<code>total_gold</code>	<code>goldEarned</code>	<code>Total_gold</code>
<code>dpm</code>	Not provided	<code>Dpm</code>
<code>gpm</code>	Not provided	<code>Gpm</code>
<code>vision_score</code>	<code>visionScore</code>	<code>Vision_score</code>
<code>gold_diff_10</code>	Not provided	<code>Gold_at_10</code>
<code>xp_diff_10</code>	Not provided	<code>Xp_diff_at_10</code>

This normalization ensures that metrics with identical meaning share the same field name, enabling direct comparison across both datasets.

3.3 Derived Metrics

SoloQ JSON data and the professional match dataset expose different raw fields and event structures. To compare gameplay outcomes across these two sources within a unified schema, several derived metrics were constructed. These metrics normalize the data into a shared analytical space where identical concepts have the same meaning regardless of how they were originally recorded.

All derived metrics follow two design rules: the value must exist or be computable in both datasets, and the metric must represent an interpretable gameplay outcome, not a file-specific field. The metrics represent five aspects of gameplay that are consistently expressed across both data sources: **combat efficiency, resource efficiency, lane outcome, vision control, and objective contribution**. For each aspect, a representative metric was derived:

3.3.1 Damage per Minute (DPM) — Combat Efficiency

DPM expresses how effectively a player converts time into combat impact. It captures trading decisions, fight timing, and damage consistency. This value can be obtained directly from Pro statistics and computed from SoloQ timeline frames.

3.3.2 Gold per Minute (GPM) & CS per Minute (CSPM) — Resource Efficiency

GPM measures how quickly a player collects and converts resources into outcomes. CSPM isolates lane farming ability by measuring minion control and wave timing. Together, these two metrics express the player's rate of income through micro-level decisions (CS control) and macro-level actions (objective rewards).

3.3.3 Lane Pressure Index — Lane Outcome

To summarize early lane state in one value, the project uses an index based on gold, XP, and CS differences at minute 10. This metric indicates whether the player created early pressure that affects rotations, matchup control, and mid-game objective timing.

3.3.4 Vision Efficiency — Vision Control

Vision efficiency normalizes ward placement, clearing, and score contribution by time, indicating how well a player uses vision tools to support movement safety and objective preparation.

3.3.5 Team Dragons — Objective Contribution

Objective participation is represented through team dragon control, which requires macro coordination, map pressure, and timing choices. This metric captures contribution that is not visible from individual combat statistics alone. These derived metrics enable the unified table to express the same gameplay concepts for both sources even when the underlying fields differ.

All definitions, variable names, and calculation rules are documented in `metadata.json` following the `schema.org` Dataset model, and each value can be reproduced through the curation scripts in this project.

3.4 Observation Unit and Aggregation

In the unified dataset, each observation corresponds to a **single player's performance in a single match**. This row-level structure is consistent across SoloQ JSON data and professional CSV data, after removing all personal identifiers and harmonizing metrics.

However, row-level values alone do not provide meaningful insights. Since League of Legends performance varies significantly by patch version, role, and skill level, aggregated statistics are computed along the following dimensions:

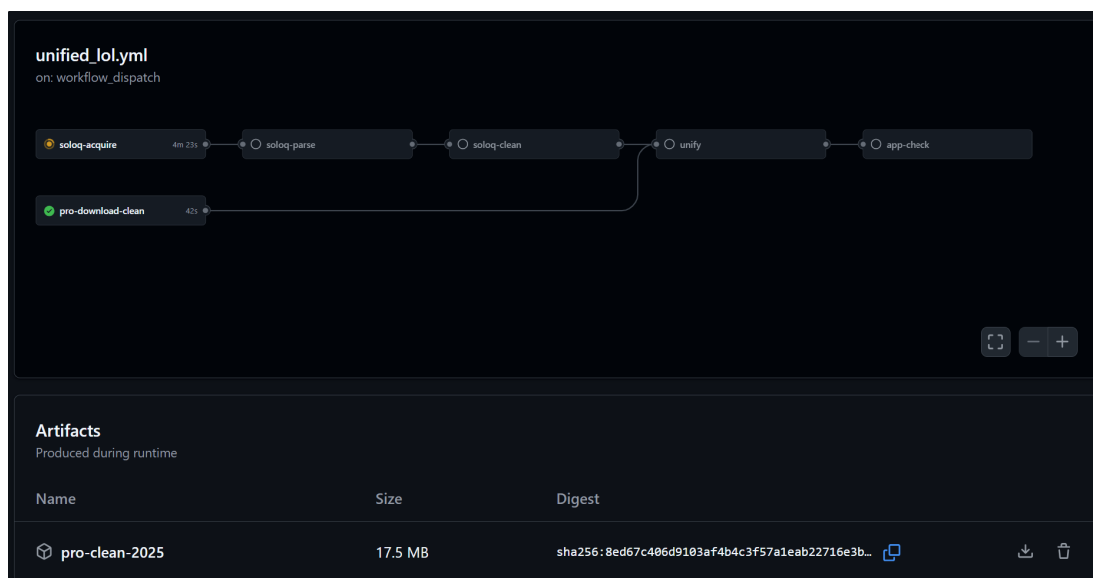
- **Patch version:** Both datasets include a match-level patch field. Aggregating observations by patch ensures that comparisons are not distorted by meta changes.
- **Dataset type (SoloQ vs Professional):** Allows controlled comparison between amateur and competitive play.
- **Role (TOP, JGL, MID, ADC, SUP):** Normalizes differences that originate from lane-specific responsibilities.
- **Tier (Iron → Challenger):** Enables analysis of skill progression and how closely each tier resembles professional behavior.

Thus, while each row represents a single player in a single match, all insights presented in the visualization dashboard are generated by grouping rows according to **patch × dataset × role × tier**, enabling stable and interpretable comparisons across skill levels and competitive environments.

3.5 Data Processing Pipeline

The workflow follows the USGS Science Data Lifecycle Model. GitHub Actions collect new data on a schedule and keep all versions of the dataset. A schema.org Dataset metadata file was created to describe the data clearly. It includes variable names, definitions, file structure, provenance, and license information in Appendix B. A [workflow.yml file](#) automates the entire pipeline in a clean Python environment, installing dependencies, executing each stage, and storing intermediate results as artifacts. This design provides a traceable lineage of the dataset and ensures reproducibility across acquisition, cleaning, unification, and validation.

Figure 1. Github Action workflow



To keep the dataset consistent, all unified columns use snake_case, and derived metrics such as DPM, GPM, and CSPM follow fixed naming rules across both SoloQ and professional data. All naming rules and variable definitions are listed in Appendix C.

The variables in the unified dataset match the goal of comparing SoloQ and professional play. Combat metrics (DPM, KDA, CSPM) show individual skill. Economy metrics (GPM, CS) show resource control. Vision and objective metrics (vision score, dragons, barons, towers) show team coordination and macro play. These groups highlight the main differences between everyday players and professional teams.

All exploratory analysis scripts, visualizations, and comparative notebooks are provided in the [analysis directory](#) of the project repository for transparency and reproducibility. A detailed description of workflow automation, reproducibility, and dissemination is provided in the project's [README](#).

USGS Stage	Pipeline Step	Summary
Acquire	3.5.1 Acquisition (SoloQ/acquire.py)	Professional CSVs and SoloQ API data are collected through automated workflows.
Process	3.5.2 Parsing and Cleaning (SoloQ/parse.py, SoloQ/clean.py, pro/clean.py)	Raw data are validated, normalized, and cleaned.
Analyze	3.5.3 Transformation (unified.py)	SoloQ JSON is flattened, pro CSVs are aligned, and derived metrics are computed.
Preserve	3.5.4 Unified Schema (unified.py)	Both datasets are converted into a shared schema and stored in versioned form.
Publish / Evaluate	3.5.5 Reproducibility (app.py)	Outputs are regenerated through scheduled workflows and released as versioned artifacts.

3.5.1 Acquisition

To follow [Riot's rate limits](#) (**20 requests per second and 100 requests per two minutes**), the pipeline uses a fixed delay and a retry system. Each API call waits **0.7 seconds**, which keeps the request rate well below the limit.

All API calls use a helper function, `riot_get`, which handles errors safely. If the server returns a 429 (Too Many Requests) message, the script checks the Retry-After header. If the header is missing, it uses exponential backoff (1, 2, 4, 8, and 16 seconds). The script allows up to five retries, which helps recover from short network problems without stopping the whole workflow.

Server errors (5xx) and timeouts follow the same retry pattern. Client errors (4xx) stop immediately because they indicate invalid requests.

The script also avoids downloading the same file twice. Before saving a match or timeline, it checks if the file already exists and skips it if so. This makes the pipeline

safe to run again on GitHub Actions, because any failed downloads will be retried in later runs. Overall, this design handles rate limits, temporary server issues, and network errors in a simple and reliable way.

- **Professional Data:** A workflow downloads season-level CSV files from Google Drive (Oracle's Elixir exports). Users specify the target year, and the workflow filters the raw directory to retain only the corresponding season file (e.g., pro_2025.csv).
- **SoloQ Data:** Another workflow collects ranked SoloQ matches using the Riot API. The workflow accepts parameters such as Riot API key, patch prefix, and optional page limits. Rate-limit handling is built in through masked API keys, retry logic, and bounded pagination.

3.5.2 Cleaning(Process/Validate)

- Corrupted, incomplete, or remake matches are removed.
- Patch versions are normalized to consistent formats (e.g., 15.01 → 15.1).
- Personal identifiers such as player names, puuids, and team IDs are stripped.
- Missing fields (e.g., visionScore in SoloQ) are recorded explicitly as nullable values.
- Timestamps, numeric fields, and categorical roles are standardized.
- All cleaning and normalization steps are implemented in reproducible scripts located in the repository (/SoloQ/clean.py and /pro/clean.py), and no manual editing was used.

3.5.3 Transformation(Process/Analyze)

- SoloQ JSON files are flattened into participant-level rows, and lane and experience differentials around the early game (e.g., at 10 minutes) are extracted from Riot's challenges fields rather than raw timeline frames.
- Professional CSV files require column renaming and alignment to match SoloQ variable semantics.
- Derived metrics—including DPM, GPM, kill participation, lane differentials, vision efficiency, and objective participation are computed to ensure consistent definitions across datasets.

3.5.4 Integration into a Unified Schema(Analyze/Preserve)

- Both datasets are converted into a shared participant-match schema with harmonized field names and consistent definitions.
- Each row is annotated with patch version, dataset type, role, and tier to support structured aggregation.

- The unified dataset is stored in a stable, versioned format suitable for tier-level, role-level, and SoloQ-Pro comparisons.

3.5.5 Reproducibility(Preserve/Publish/Evalute)

- Automated scheduled execution
- Versioned artifacts
- Regeneration of the unified dataset when new patches or seasons are released

3.6 Data Abstraction

This project combines SoloQ JSON data and professional match CSV data into one unified dataset. Three data structure ideas are used: tree structure, relational table structure, and a shared definition of metrics. Each idea matches a real step in the data process.

3.6.1 Tree structure (JSON)

SoloQ match data from the Riot API is nested JSON. A match is the root, and it contains metadata, participant data, timeline frames, and event logs. This structure works the same as a tree. The data is flattened so that each participant becomes one row without nested layers.

3.6.2 Relational Structure (Unified Table)

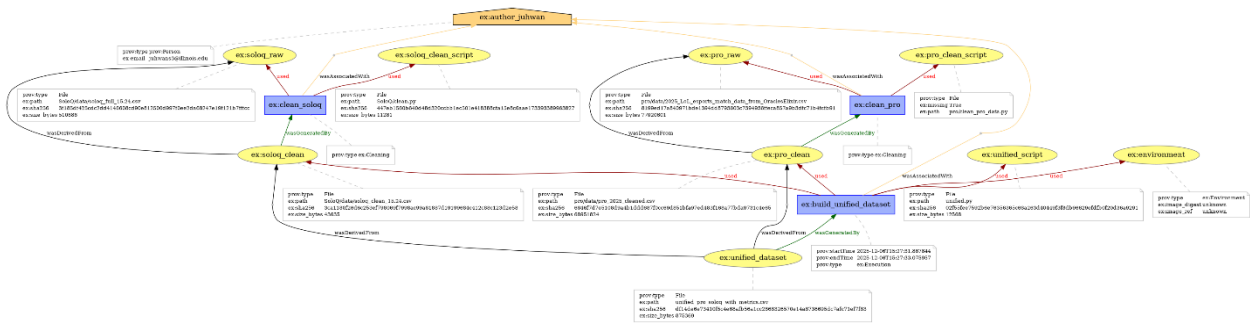
Once flattened, both SoloQ data and pro match data follow the same column layout. Fields with the same meaning are mapped to one column, such as SoloQ goldEarned and pro total_gold. This produces a single table format that can be used for analysis across ranks and seasons.

3.6.3 Shared Metric Definitions

Some metrics require a shared meaning, not just matching field names. Values like Damage Per Minute and Gold Difference at 10 minutes are defined with one clear concept. These definitions are not tied to file format.

The definitions and basic dataset description are stored in [metadata.json](#) using the schema.org Dataset format. The overall process of creating the dataset is recorded in [provenance.json](#), and a diagram of the same information is provided in [provenance.png](#).

Figure 2. provenance.png



3.6.4 Ontology/Graph Abstraction

The project uses a graph-based ontology through the W3C PROV model to describe data lineage. Unlike tree and relational abstractions, the provenance graph expresses many-to-many semantic relationships between datasets, processing activities, and agents. Raw SoloQ JSON and raw Pro CSV are modeled as entities, acquisition/cleaning/unification steps are modeled as activities, and data providers and the GitHub workflow are modeled as agents. This ontology shows how the unified dataset is derived from its sources, which transformations were applied, and how each artifact is semantically connected.

3.7 Reproducibility

The full curation pipeline is executed end-to-end using [GitHub Actions](#). Each step of the workflow, data acquisition, parsing, cleaning, and unification is defined as an automated job, and no manual file edits or intermediate processing are required. The workflow takes two inputs and writes all intermediate datasets as versioned GitHub artifacts to support transparent reruns and debugging.

4. Visualization and Insights

[The dashboard](#) is designed to clearly reveal how gameplay behaviors vary across tiers and how closely each group aligns with professional standards. All visual components focus on role-specific performance, early-game outcomes, and strategic decision-making.

4.1 Dashboard Structure

The dashboard focuses on tier-based differences and role filters rather than complex interactive panels.

- **Summary Counters:** At the top of the page, the app shows the total number of rows, the number of SoloQ rows, and the number of professional rows for the current role selection.
- **Role Filter:** A sidebar control lets the user select one or more roles. All visualizations are updated based on the chosen roles.

- **Tier Progression Line Charts:** For a fixed set of key metrics (DPM, GPM, CSPM, vision efficiency, team dragons, and lane pressure index), the app draws line charts that show the mean and standard deviation of each metric by tier, including the PRO tier.
- **Tier-wise Distribution Boxplot:** A second view lets the user select a single metric and see its distribution across tiers as a boxplot with all points visible. This helps compare how spread out or stable each metric is across ranks.

4.2 Insights

Using these visualizations, it is possible to see several consistent patterns in the data:

- Early-game lane metrics, such as gold, XP, and CS differences at 10 minutes, tend to improve as tier increases, and higher tiers move closer to the professional tier.
- Combat and resource metrics, such as DPM and GPM, also rise with tier, which suggests that players at higher skill levels use gold and experience more efficiently.
- Vision-related metrics and objective-related metrics can be inspected through the tier progression and boxplots, and they often show noticeable gaps between SoloQ tiers and professional games, especially in lower tiers.

The dashboard does not compute automatic conclusions, but it gives a clear view of how key metrics change by tier and makes it easy to visually compare SoloQ ranks with the professional tier.

5. Limitations

This project has several practical limitations caused by differences in data formats, limited data access, and patch timing in League of Legends.

5.1 Structural Differences Between SoloQ and Professional Data

SoloQ data includes minute-level timeline frames, so early-game changes can be tracked in detail.

Professional data, however, only provides checkpoints at 10, 15, 20, and 25 minutes. Because of this:

- Some metrics (such as lane pressure or jungle tempo) can be measured precisely in SoloQ but only estimated in the pro dataset.
- Early-game metrics must be normalized to the lowest shared resolution, which reduces overall detail.

This difference is built into the data sources and cannot be fully removed.

5.2 API and Data Availability Constraints

SoloQ data depends on the Riot API, which has strict rate limits (20 requests per second and 100 requests per 2 minutes). This slows down large-scale data collection and may cause sampling bias across tiers or dates.

Professional data has a different challenge. It is not available through an API, and must be downloaded manually from Google Drive provided by Oracle's Elixir contributors.

I attempted to automate the download using crawling tools, but Google Drive blocks scripted download patterns for large files or external clients. Because of these restrictions, automated collection was not possible, and this part required manual work. This made the pipeline slower and harder to fully automate.

5.3 Patch Misalignment Between Professional and SoloQ Play

Patches strongly affect champion balance and strategy. SoloQ players switch to new patches immediately, but professional leagues often stay one patch behind because they need time for competitive testing. Because of this difference:

- SoloQ and pro players may use different versions even within the same week.
- Even when the dashboard aligns matches by patch number, the real-world timing may still be different, which reduces direct comparability.

5.4 Coverage and Sampling Bias

Match volume is very different across tiers. Some groups, such as Master+ players or rare roles have fewer samples, which can cause high variance or unstable averages. Pro data also has limited match volume because a competitive season produces far fewer games than SoloQ.

6. Ethical, Legal, and Policy Considerations

SoloQ data collected from the Riot API includes personal identifiers such as puuid and summoner names. These identifiers were removed to follow Riot's Terms of Use and to prevent re-identification. [The Riot API also has strict rate limits](#) **20 requests per second and 100 requests every two minutes**, so the pipeline uses delays and limited retries to stay within these limits.

Professional match data from Oracle's Elixir is shared through community-maintained Google Drive files and usually does not include clear redistribution rights. Because of this, the project does not publish any raw pro files and uses only derived metrics and unified variables for analysis.

Professional play also uses patches later than SoloQ. Even if the patch numbers look the same, the timing may be different, and this affects how players choose

champions or items. This patch delay was considered when comparing ranked and pro matches.

The project uses several identifiers to keep data tracking clear. **matchId** is used to follow each game through acquisition, parsing, and integration. Player identifiers like **puuid** are removed in the final dataset for privacy but were kept briefly during processing to check role and lane mapping. Versioned filenames and patch-based folders provide a simple way to preserve provenance between raw, cleaned, and unified data.

7. Future Work

Future extensions will focus on shifting the analysis from tier-level summaries to **personalized insights**. While the current system compares aggregated SoloQ tiers with professional benchmarks, the next stage will integrate individual player data to quantify how a specific user plays relative to both their tier and professional standards.

(1) Personalized Performance Modeling

A major goal is to compute a user's own metrics—combat, laning, vision, movement, and objective patterns—and position them within the unified statistical space. Using timestamp-based SoloQ timelines, the system can highlight how the user's decisions differ from higher tiers. For example:

- "Your early roaming frequency matches Silver MID averages, while Diamond players exhibit more structured rotations."
- "Your warding efficiency aligns with Gold-tier supports but remains below professional benchmarks."

This enables players to understand their specific gap and what aspects lead to higher-tier behavior.

(2) Player-Pro Similarity Analysis

To make the experience more engaging, future versions will include similarity modeling that identifies which professional players a user most closely resembles. This requires computing per-player embeddings for professionals and comparing them with individual SoloQ embeddings. Examples include:

- "Your playstyle is most similar to **Chovy** among current pro mids."
- "Your objective timing resembles **Oner**'s early-game pathing."

This direction combines statistical profiling with lightweight machine learning to provide intuitive, motivational insights.

(3) Patch-Aware Strategy Guidance

League of Legends changes significantly with each patch, and professional play typically lags behind SoloQ. Future versions will incorporate patch metadata to evaluate how a user's champion pool and playstyle interact with the current patch trends.

For example:

- "This patch favors lane-dominant mid champions; based on your historical data, your current pool benefits from this shift."
- "Your champion's win rate increased after the latest patch; the matchup patterns are now more favorable."

Integrating patch-level win rate, pick/ban rates, and item changes will enable dynamic, patch-aware recommendations.

7. Conclusion

This project curated and integrated two structurally different data sources—SoloQ timeline data and professional match statistics—to build a unified framework for comparing gameplay behavior across skill levels. By harmonizing JSON-based minute-level SoloQ data with CSV-based professional datasets, a consistent schema was created that allows direct comparison of combat metrics, laning outcomes, vision control, objective participation, and efficiency indicators.

The resulting visualization dashboard highlights how player behavior evolves across tiers and how closely each group aligns with professional benchmarks. Through role- and tier-specific summaries, comparative radar charts, and progression graphs, the system reveals clear differences in strategic decision-making, early-game discipline, and map control between ranked play and professional competition.

While the current work provides an interpretable, group-level view of gameplay patterns, its structure enables natural extensions toward personalized insights. Future iterations can incorporate individual player timelines, pro-player similarity modeling, and patch-aware guidance to offer users a deeper understanding of their playstyle and improvement pathways.

Overall, this project demonstrates how careful data curation—spanning acquisition, normalization, schema design, and visualization—can transform complex multi-source esports data into a coherent analytical tool for exploring gameplay behavior and bridging the gap between everyday players and professional standards.

8. References

- Riot Games. (n.d.). *Riot Developer Portal*. Riot Games. Retrieved September 15, 2025, from <https://developer.riotgames.com>

- Riot Games. (n.d.). *Riot Games Developer Terms of Use*. Riot Games. Retrieved November 30, 2025, from <https://developer.riotgames.com/terms>
- Riot Games. (n.d.). *General API Policies*. Riot Games. Retrieved November 30, 2025, from <https://developer.riotgames.com/policies/general>
- Oracle's Elixir. (n.d.). *Professional League of Legends match data*. Retrieved September 15, 2025, from <https://oracleselixir.com/>
- Leaguepedia. (n.d.). *Leaguepedia API*. Retrieved September 15, 2025, from https://lol.fandom.com/wiki/Leaguepedia_API
- schema.org. (n.d.). *Dataset*. Retrieved September 15, 2025, from <https://schema.org/Dataset>
- DataCite Metadata Working Group. (2021). *DataCite Metadata Schema for the Publication and Citation of Research Data (Version 4.4)*. DataCite e.V. <https://schema.datacite.org>
- World Wide Web Consortium. (2013). *PROV-Overview: An overview of the PROV family of documents (W3C Recommendation)*. W3C. <https://www.w3.org/TR/prov-overview/>

Appendix

Appendix A Tables

- Table A1. Derived Metrics Used in the Unified Schema

Metric Name	Formula / Source	Applies To	Description
Damage per Minute (DPM)	SoloQ: $\text{totalDamageDealtToChampions} / (\text{duration_sec}/60)$ / Pro: provided (dpm)	Both	Measures damage output efficiency independent of game duration.
Gold per Minute (GPM)	SoloQ: $\text{goldEarned} / (\text{duration_sec}/60)$ / Pro: provided (earned_gpm)	Both	Indicates economic efficiency and farming speed.
Vision Score per Minute (VSPM)	$\text{visionScore} / (\text{duration_sec}/60)$	Both*	Measures map control contribution; SoloQ missing → nullable.
Kill Participation (KP)	$(\text{kills} + \text{assists}) / \text{teamkills}$	Both	Quantifies teamfight involvement and teamwork level.
Damage Share	$\text{damage_to_champions} / \text{team_damage}$	Both	Measures relative contribution to team's total damage.

Gold Share	$\text{total_gold} / \text{team_total_gold}$	Both	Indicates player's share of team resources.
Resource Conversion Efficiency (RCE)	DPM / GPM	Both	Measures how efficiently a player converts gold into damage.
Objective Participation Rate (OPR)	$\text{objectives_participated} / \text{team_objectives}$	Both	Measures involvement in dragons, heralds, and barons.
Lane Pressure Index (LPI)	$(\text{golddiff} + \text{xpdiff} + \text{csdiff}) / 3$	Both	Captures lane dominance during early laning phase.
Gold Diff @10	SoloQ: timeline-derived / Pro: golddiffat10	Both	Early-game gold advantage at 10 minutes.
XP Diff @10	SoloQ: timeline-derived / Pro: xpdiffat10	Both	Early-game XP advantage at 10 minutes.
CS Diff @10	SoloQ: timeline-derived / Pro: csdiffat10	Both	Early-game CS advantage at 10 minutes.
Aggression Index	$(\text{kills} + \text{assists}) / (\text{deaths} + 1)$ OR $\text{damage_to_champions} / \text{damage_taken}$	Both	Measures risk-taking and fighting frequency.
Vision Efficiency	$(\text{wards_placed} + \text{wards_killed}) / \text{duration_min}$	Both	Captures warding and vision clearing effectiveness.
Gold Spending Efficiency (GSE)	$\text{gold_spent} / \text{total_gold}$	Both	Measures how efficiently gold is converted into completed items.
First Action Time (FAT)	timestamp(first kill/death/assist)	SoloQ only	Indicates how quickly the player becomes active in the game.

Appendix B. Dataset Metadata (schema.org JSON-LD)

- The full metadata file is available in the project repository:
https://github.com/ddori/LeagueOfLegends_Data_Curation/blob/main/meta_data.json

Appendix C. Naming Standards and Provenance Artifacts

C.1 Naming Rules Used in This Project

To maintain consistency across SoloQ JSON data and professional CSV data, the project applied clear naming rules at three levels:

Metadata naming (schema.org Dataset)

- Uses camelCase naming as required by the schema.org Dataset vocabulary.
- Examples: name, description, variableMeasured, distribution.

Dataset column naming

- Uses snake_case for all unified dataset columns.
- Ensures SoloQ and professional data follow the same style even when original sources use different conventions.
- Columns: total_damage, dpm_share, gold_diff, lane_position.

Derived metric naming

- Uses fixed prefixes and standard definitions so meaning is consistent across patches.
- DPM = damage / minutes
- GPM = gold / minutes
- OPR = objective participation rate
- Naming rules ensure stable interpretation during integration and analysis.

C.3 Provenance Artifacts (GitHub Actions)

The automated acquisition and parsing steps are executed through [GitHub Actions](#) workflows. A visual snapshot of the workflow graph is included below to document provenance for the data acquisition process.

Appendix D. Data Dictionary (Unified SoloQ-Pro Dataset)

In the unified dataset, both SoloQ and professional matches share the same schema based on the pro dataset. The “Origin” column indicates whether each variable is taken directly from the raw match files (“raw”) or computed in this project (“derived”). SoloQ rows are mapped into the same schema; if a value is not available for a given source, it appears as NA.

Variable	Type	Unit	Origin	Missing rule	Description
matchId	string		raw	NA	Unified match identifier (Riot ID or pro ID mapped to a common format).
patch	string		raw	NA	Game patch (e.g., 15.20).
duration	float	minutes	raw	if ≤0, set to 1	Game length in minutes.

role	string		raw	NA	Player role (top, jungle, mid, adc, support).
champion	string		raw	NA	Champion played.
kills	int		raw	NA	Number of kills.
deaths	int		raw	NA	Number of deaths.
assists	int		raw	NA	Number of assists.
kda	float		derived	if deaths = 0 → kills + assists	KDA ratio.
total_gold	int	gold	raw	NA	Total gold earned.
gpm	float	gold/min	derived	if gold or duration missing → 0	Gold per minute.
dpm	float	dmg/min	raw/derived	if damage or duration missing → 0	Damage per minute (direct for pro; computed for SoloQ).
cs_total	int	CS	derived	if components missing → 0	Lane CS + jungle CS.
cspm	float	CS/min	derived	if CS or duration missing → 0	CS per minute.
visionScore	int		raw	NA	Vision score.
team_dragons	int		raw	NA	Dragons taken by the player's team.
team_barons	int		raw	NA	Barons taken by the player's team.
team_towers	int		raw	NA	Towers taken by the player's team.
win	bool		raw	NA	Game result (win or loss).