Roll Initiative!

A Data-Driven Analysis of First-Mover Advantage in D&D 5E Rules

Clara W.

February 2026

Table of Content

I. Executive Summary

"... Roll initiative!" says your dungeon master.

This single moment often decides the tempo of everything that follows in turn-based games. It begins the same way: a dice roll, a number, to determine the action order. Acting first feels powerful. However, the real impact of initiative is difficult to isolate through playtesting because individual encounters are highly variable and influenced by human decision-making.

To address this, this project was developed to run thousands of repeatable combat encounters through a controlled simulation model under fixed rules. The outcomes would be analysed and presented with SQL, Python, and Tableau.

Across 5,000 simulated encounters, comparing three different classes, results show that acting earlier has a positive correlation with win rate. However, the relationship is not linear or purely beneficial. The findings suggest that initiative advantage operates through a dynamic interaction between early burst damage, targeting rules, and survivability trade-offs, especially for low-HP characters.

Data Sources

Static monster and equipment data were obtained from the Kaggle dataset "Dungeons & Dragons" (shadowtime2000), which aggregates structured data derived from the D&D 5e API (https://www.dnd5eapi.co/), licensed under the MIT License.

Only fields relevant to combat simulation (e.g., AC, hit points, attack bonuses, damage dice) were selected and curated for this project.

II. Problem Statement and Proposed System

Problem statement

In turn-based combat systems, acting earlier often feels powerful. However, it is not

always clear how much initiative really matters. In D&D, initiative is decided by a random 1~20 dice roll; a slower character can act first and a faster character can act late.

So I am particularly interested in these questions:

- Does acting earlier increase the chance of winning?
- How much damage can be dealt before the opponent can respond? And how much is it correlated with winning?
- How often does randomness override character advantages?

These questions are difficult to answer through playtesting alone, because individual encounters are noisy and inconsistent and players' decision-making also introduces bias. So a data-driven approach could be the best way to isolate these effect of initiative.

Proposed system

To address this problem, this project builds a simulation-based analysis system to:

- Simulate more than 2000 of combat encounters.
- Use fixed rules and fixed character templates.
- Log every encounter and participant into a SQLite database.
- Analyze outcomes using SQL queries and visual dashboards.

System overview

The proposed system follows a simple pipeline:

- Curated data is loaded from Kaggle datasets (CSV files) into a SQL database.
- An encounter template defines the participants.
- A simulation python script runs the combat many times.
- Each run records initiative order, damage, and outcomes.
- Results are queried and analyzed using SQL and Python.
- Key findings are visualized in Tableau.

Expected outputs

The system produces combat logs that can calculate:

- Win rates based on initiative advantage.
- Survival rates by initiative order.
- Damage dealt before the first opponent turn.
- Comparisons between early-acting and late-acting participants.

III. Scope & Assumptions

Scope

This project models a simplified combat system inspired by D&D 5e. To narrow down to the initiative order and combat outcomes, the settings and rules are simplified.

The scope includes:

- A fixed party of three characters at Level 3.
- A fixed enemy encounter (4 Goblins and 1 Bugbear).
- Initiative resolution using a d20 roll plus a modifier.
- Basic attack actions only.
- Critical hits and damage rolls.
- A small opening-round features (Rogue's opening advantage) that increase early damage.
- Large-scale simulation (thousands of repeated runs).
- Storage of results in a SQL database for analysis.
- Analysis using Pandas, SQL, and Tableau.

Out of scope

The following elements from D&D 5E rules are intentionally excluded:

- Movement, positioning, and terrain.
- Reactions and opportunity attacks.
- Most bonus actions.
- Death saving throw.
- Conditions (e.g. prone, stunned, poisoned).
- Full spell systems and saving throws.
- Stealth, surprise, and visibility mechanics.
- More weapon options for each participant.
- Dynamic or adaptive AI decision-making.
- Variable party compositions or encounter types (now in ver. 1).

To avoid scope creep, the following decisions are locked for version 1:

- No additional class features beyond those already defined.
- No new enemy types or encounter layouts.
- No real-time or interactive gameplay.

Assumptions

The simulation is built on the following assumptions:

- Results are **scenario-specific** and should not be treated as general game balance conclusions.
- Initiative matters mainly through early access to damage.
- A small number of burst-style features is enough to represent first-mover advantage.
- Fixed targeting rules create more consistent and comparable results.
- Simplifying combat rules does not change the direction of initiative effects, even if it changes exact win rates.

- Randomness from dice rolls averages out at high simulation counts.

The system is designed to focus on initiative rather than full tactical realism, and possible for future update or ver. 2 of combat.

IV. Constraints and risks

Constraints (known limits of the project)

Model complexity constraint

- Intentionally excludes movement, positioning, conditions, reactions.
- Most class features are excluded.

Data source constraint

- Dice data are sourced from static CSV files, adding complexity to the contextual abilities. (e.g. 1d8+3 is obvious for human, which means dice_count = 1, dice_sides = 8, modifier = +3)

Reproducibility vs realism trade-off

- Following fixed party composition, fixed targeting rules, and frozen tie-breaker logic.
- Trade-off: reduced tactical variability.

Risks (uncertainties and how they're managed)

Over-interpretation risk

- Results may be misread as general D&D balance conclusions rather than analysis specific to this encounter and rule set.
  Mitigation: Framing the system as *scenario-bound* and clarifying with a limitations section and possible v2 roadmap.

Model bias risk

- Simplified targeting logic (e.g. PCs always focus the Bugbear first, monsters target lowest HP) may amplify early damage snowball effects.

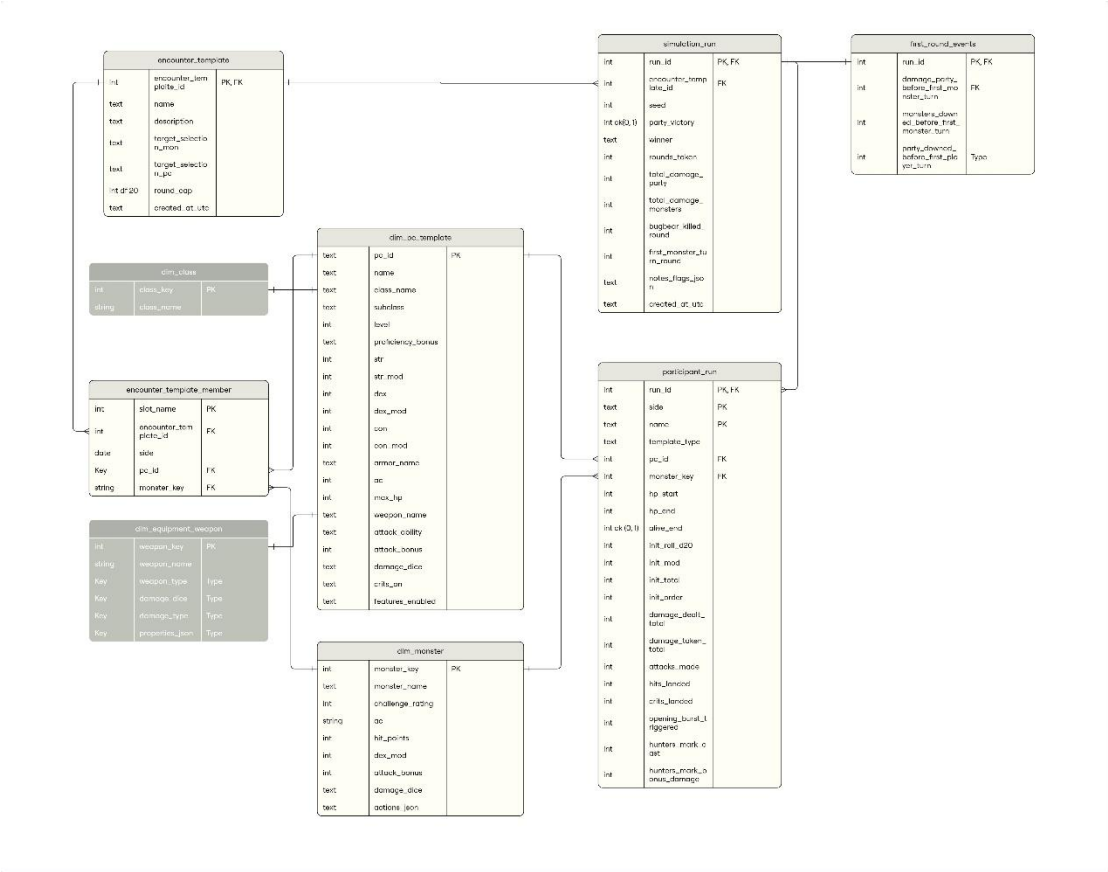
Mitigation: Targeting rules are explicitly held constant across all simulations for comparison.

Sampling variance risk

- Smaller run counts could produce misleading win rates due to randomness in dice rolls.
  Mitigation: Simulations are run at scale (5,000–10,000+ iterations), and observed rates are compared against expected probability ranges.

V. Data model and entity definition matrix

An entity relationship model was developed during the design stage of the database:



The model separates scenario setup from simulation results: encounter_template and encounter_template_member define who participates and the flow for selecting targets, while simulation_run stores one complete simulation outcome (win/loss, rounds, total damage).

Performance is captured at two levels: participant_run records per-entity metrics inside each run (initiative order, damage dealt/taken, hits/crits), and first_round_events stores early-phase indicators such as “damage before the first opponent turn” to measure early advantage.

Reference tables, including dim_pc_template, dim_monster, dim_equipment_weapon, dim_class, standardize attributes so reporting can group results by build, opponent type, and equipment without duplicating data.

Below is the detail explanation for each table:

Entity / Table	Definition	Key fields (PK/FK)	Why it exists
encounter_template	A reusable “scenario blueprint” describing how an encounter should be constructed (targeting rules, caps, metadata).	PK:encounter_template_id	To compare outcomes across different encounter setups: “Does encounter A have higher win rate than encounter B?” “Do different selection rules change results?”
encounter_template_member	The specified roster for a template: which slots are PCs vs monsters, and which specific PC template or monster is used.	PK: slot_name (plus template scope) FK: encounter_template_id → encounter_template, pc_id → dim_pc_template, monster_key → dim_monster	Specifies the participants it has in a run.
simulation_run	One full simulated battle outcome for one	PK: run_id FK:	To support most top-line analysis:

	encounter template under one random seed. The run-level “headline metrics”.	encounter_template_id → encounter_template	party win rate, rounds_taken, total_damage_party/ monsters, “Rogue beats Bugbear?”, “How often does the party win before round cap/timeout?”
participant_run	One row per participant per run: stores initiative roll/order and performance totals (damage dealt/taken, hits/crits, feature flags).	PK: run_id, side, name FK: run_id → simulation_run, pc_id → dim_pc_template, monster_key → dim_monster	Enables micro-analysis, like “Does higher initiative order correlate with higher win rate?” “Which participant contributes most damage?” “Do crit features (ex: champion 19–20) show up as higher `crits_landed`?” Also supports “opening burst” / “hunter’s mark” impact.
first_round_events	Run-level alpha-strike snapshot: what happened before the first monster turn (damage + downs).	PK: run_id FK: run_id → simulation_run	Directly supports: “Damage before first monster turn vs win rate” and “How often do monsters get downed before acting?”
dim_pc_template	Canonical definition of a PC build used in	PK: pc_id	Lets you slice results by PC

	simulations (stats, AC/HP, weapon + attack profile, crit rules, feature flags).		build: “Does Rogue build outperform others?” “Which PC template drives most early damage?” Also prevents duplicating build details in every participant_run row.
dim_monster	Canonical monster definition (CR-ish data, AC/HP, attack and actions).	PK: monster_key	Supports “Which monster types swing outcomes?” and “Does Bugbear specifically get killed earlier?” Also keeps monster stats consistent across runs.
~~dim_equipment_weapon~~	~~Canonical weapon definitions used by PCs/monsters (type, dice, damage type, properties).~~	~~PK: weapon_key~~	~~Supports analysis like: “Do certain weapons correlate with higher damage_dealt_total?” and keeps weapon rules/data centralized (instead of hardcoding everywhere).~~
~~dim_class~~	~~Optional normalization for class names/keys.~~	~~PK: class_key~~	~~Useful when reporting: “By class, which has highest win~~

			~~contribution?".~~

The last two table, dim_equipment_weapon and dim_class, are cancelled during the later developing stage to simplified the simulation process and data complexity.

VI. Code structure and file guide

What each file does

SQL

- sql/schema.sql
  Creates all database tables and constraints (dimensions + fact tables).
- sql/analysis_queries.sql
  Saved queries used to answer the research questions (win rate, survival, early damage, etc.).

Database setup

- src/bootstrap_new_db.py
  Creates a fresh SQLite database and applies schema.sql.
- src/db_healthcheck.py
  Quick checks that tables exist and row counts look sane (useful after ETL and after simulation).

ETL (load & clean CSVs into dimension tables)

- src/etl_load_pc_templates.py
  Loads party templates from CSV into the DB (first-time load).
- src/etl_load_pc_templates_upsert.py
  Same goal as above, but supports re-running safely (updates existing rows instead of duplicating).
- src/etl_load_monsters_goblin_bugbear.py
  Loads the monster templates needed for the v1 encounter (Goblins + Bugbear).

Encounter setup

- src/seed_encounter_members.py
  Creates the fixed encounter template (3 PCs vs 4 Goblins + 1 Bugbear) and inserts encounter members.

Simulation

- src/simulate_combat.py
  Runs N simulations (e.g., 5,000+) and writes results into:
  - simulation_run (one row per battle)
  - participant_run (one row per participant per battle)
  - first_round_events (early damage / early downs)

Analysis & visuals

- tableau/dashboard.twbx
  Tableau workbook using the SQLite DB (or extracted CSVs) to build final visuals.

How to run (recommended order)

1. **Create database**: Run bootstrap_new_db.py
2. **Load dimension data:** Run ETL scripts (PC and monster templates)
3. **Seed the encounter:** Run seed_encounter_members.py
4. **Run simulations**: Run simulate_combat.py (set NUM_RUNS = 5000 or higher)
5. **Validate and analyze:**
   - Run db_healthcheck.py
   - Run analysis_queries.sql

VII. Results and analytics

The simulation results confirm that initiative order positively correlates with win rate, but they also reveal that victory emerges from interacting factors rather than acting order alone.

Party's early turns does increase damage before monsters act, which improves overall outcomes. However, the effect depends on a chain of interactions: early turn order influences early burst damage, and then determines the survival rate, total party damage and the outcome ultimately.

The simplified targeting rules also introduce a survivability trade-off: high-damage, low-HP characters such as the Rogue are frequently eliminated early. When the Rogue acts first and triggers burst damage, party win probability increases. When the Rogue acts late, early elimination reduces party damage output and lowers success rates.

A similar pattern is demonstrated by the goblins, who are targeted with a sequential order. The first-targeted goblin contributes to notably lower damage compared to the last-targeted dues to early elimination.

Meanwhile, the Fighter provides consistent damage across most simulations, contributing stability regardless of initiative variation.

The outcome data is presented by research topics and questions (Numbers are rounded to the second decimal place):

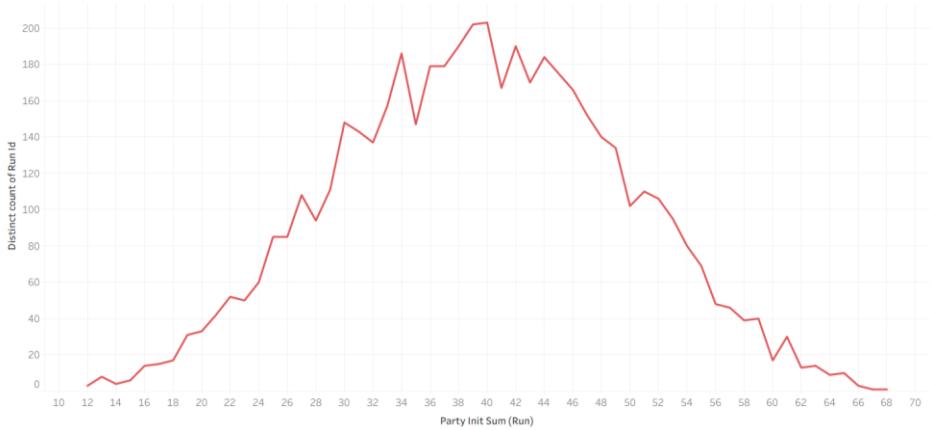
1. **Outcome distribution**
    - Party victory vs. monster victory among 5000 simulations

Winner	Run_time
Party	3036
Monster	1964

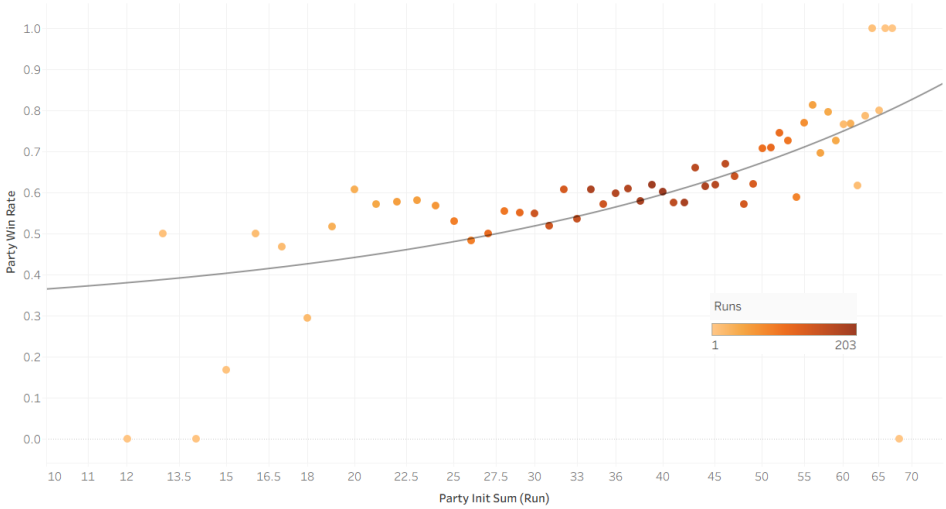
    - Win rate: party-favored but swingy

Party_win_rate
60.72%

    - Party initiative sum: initiative shows a stable, bell-shaped distribution with a mean of 39.56, consistent with expected random dice rolls.



- Party average initiative total vs win rate



- Survival Rate for every participant in 5000 runs:

side	name	Survival_rate
party	Fighter	60.72%
party	Ranger	35.02%
party	Rogue	6.18%
monsters	Goblin_4	39.28%
monsters	Goblin_3	35.96%
monsters	Goblin_2	29.2%
monsters	Goblin_1	20.32%
monsters	Bugbear	11.8%

Rogue's survival rate is notably lower than most of the participant. This **may lead to the low average of total damage by rogue** (average damage per run per participant is out of scope in this project) and amplify the importance of early-stage damage.

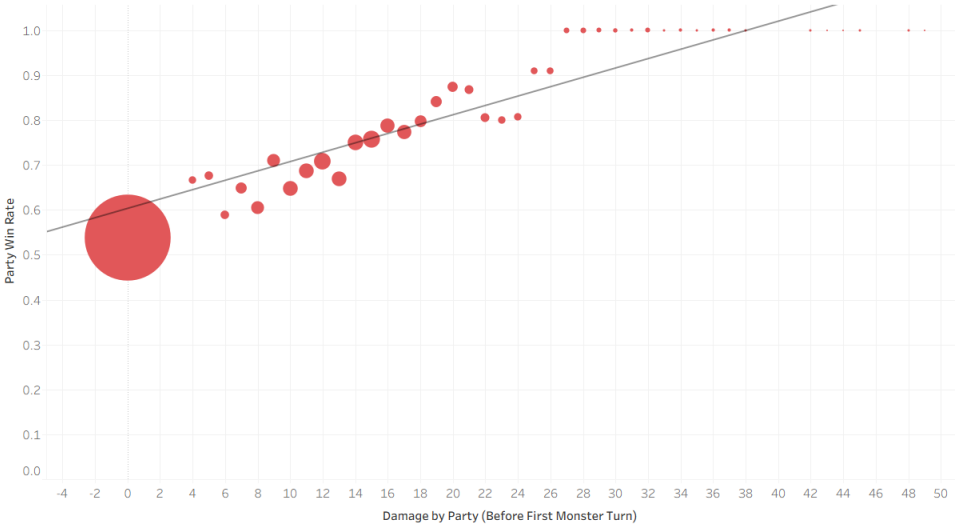
2. **Alpha-strike: damage before first monster turn vs win rate**

- Party total damage in buckets: damage and win rate show a positive

correlation.

Damage_before_ first_monster_turn	Win_rate	Runs
0	53.82%	3391
1-4	66.67%	24
5-14	67.88%	825
15-24	79.97%	604
25+	97.44%	156

However, most combats start with one of the monsters (5 monsters vs 3 PCs), so "damage before the first monster turn" is very often 0 (in 68.3% of the runs).

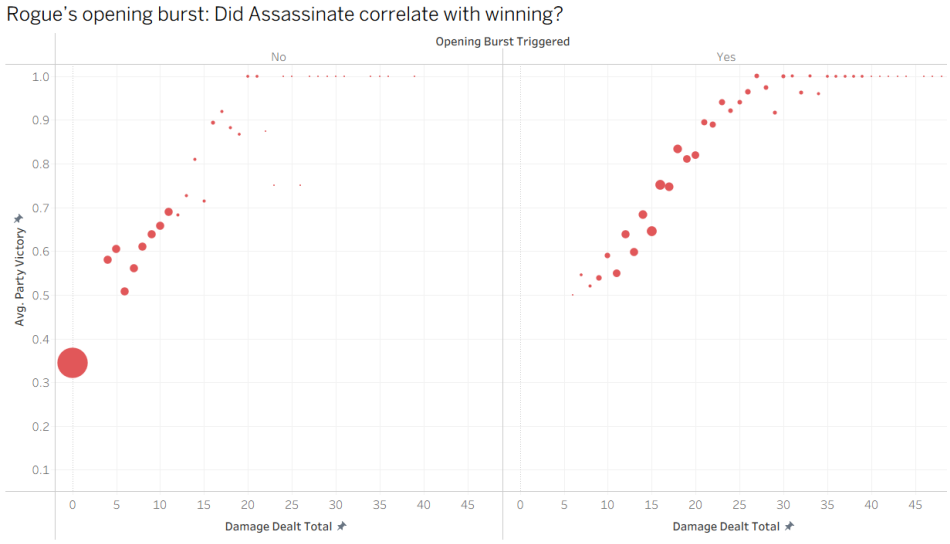


3. Rogue's opening burst: Did Assassinate correlate with winning?

- Yes; triggered burst seemed to positively correlate with party's victory. When Assassinate was triggered, party's win rate hit 77.66%, significantly more than 47.59% when Assassinate was not activate.

opening_burst_triggered	Win_rate	N
No	47.59%	2816
Yes	77.66%	2184

Assassinate also contributed to higher total damage by the party.



- However, whether the opening burst was triggered or not depended on **whether the rogue's first target (the bugbear) had attacked or not**.
- Also, does "Rogue beats Bugbear on initiative" affect win rate? There's definitely some correlation where rouge, being the first-mover, has almost 70% of the chances of winning.

rogue_beats_bugbear	Win_rate	N
No	48.73%	2079
Yes	69.26%	2921

- Rogue's average total damage when burst triggers vs not: total damage is 4 times higher when Assassinate trigged.

opening_burst_triggered	avg_rogue_damage
No	4.06
Yes	19

However, again, the precondition for Assassinate is rogue's higher initiative than bugbear's. Early actions also leads to rogue's higher survival rate and total damage.

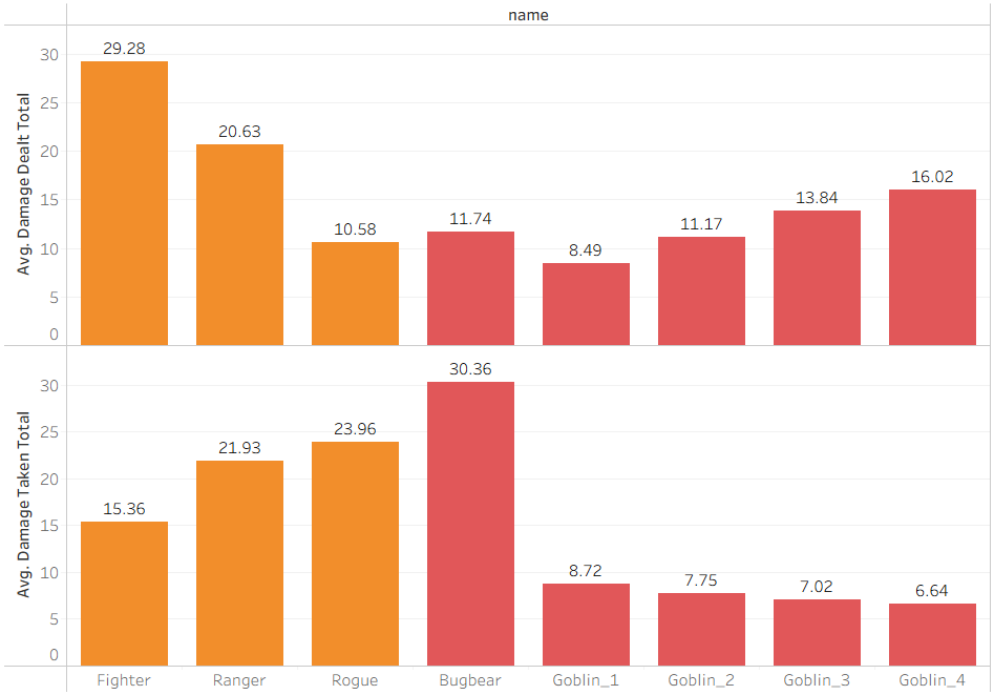
4. **Other interesting insights and comparison from the data**

- Average damage dealt per participant in 5000 runs: Fighter contributed to the party damage the most, while rogue the least. The data from goblins also reflects the low damage due to early elimination; Goblin_1, being the first target after bugbear, has significantly low damage compared to Goblin_4.

side	name	avg_damage_dealt

party	Fighter	29.28
party	Ranger	20.63
party	Rogue	10.58
monsters	Bugbear	11.74
monsters	Goblin_4	16.02
monsters	Goblin_3	13.84
monsters	Goblin_2	11.17
monsters	Goblin_1	8.49

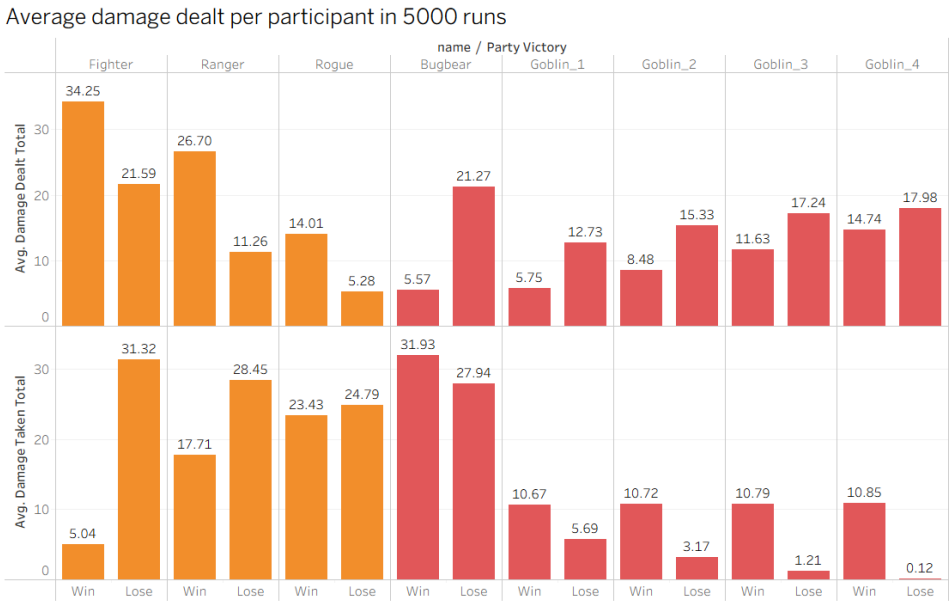
Average damage dealt per participant in 5000 runs



- Average damage dealt by participant, split by party win (3036 runs)/loss (1964 runs):

Winner	side	name	avg_damage_dealt	avg_damage_taken
party	monsters	Goblin_4	14.74	10.85
party	monsters	Goblin_3	11.63	10.79
party	monsters	Goblin_2	8.48	10.71
party	monsters	Goblin_1	5.75	10.67
party	monsters	Bugbear	5.579	31.93
party	party	Fighter	34.25	5.05
party	party	Ranger	26.7	17.71
party	party	Rogue	14.01	23.43
monsters	monsters	Bugbear	21.27	27.94

monsters	monsters	Goblin_4	17.98	0.12
monsters	monsters	Goblin_3	17.24	1.21
monsters	monsters	Goblin_2	15.33	3.17
monsters	monsters	Goblin_1	12.73	5.69
monsters	party	Fighter	21.59	31.32
monsters	party	Ranger	11.26	28.45
monsters	party	Rogue	5.28	24.79



- Party vs Monsters average total damage (per run)

avg_party_total_damage	avg_monsters_total_damage	avg_rounds
60.49	61.25	6.93

- Average hits and crits dealt per participant in 5000 runs:

side	name	avg_hits	avg_crits
party	Fighter	3.51	0.66
party	Ranger	2.11	0.20
party	Rogue	0.95	0.1
monsters	Goblin_4	2.72	0.33
monsters	Goblin_3	2.34	0.27
monsters	Goblin_2	1.89	0.21
monsters	Goblin_1	1.44	0.16
monsters	Bugbear	0.97	0.11

I noticed that rogue only makes 0.0988 critical hits per run, while fighter 0.6618 per run. It is because avg_crits does not consider the hits made by participant in a run; fighter makes way more attacks per encounter (3.5

hits in average).

To check and confirm that crit logic was accurate and as predicted, I ran the participant's crit chance per attack:

name	avg_attacks	avg_hits	avg_crits	crit_per_attack
Fighter	6.59	3.51	0.66	0.10
Ranger	4.02	2.11	0.20	0.05
Rogue	1.58	0.95	0.1	0.06

So Rogue's low crits is mostly due to volume (dies early / takes fewer turns).

- Monsters: average damage per monster type (Goblin vs Bugbear)

monster_type	avg_damage_dealt	avg_damage_taken	survival_rate
Goblin	12.38	7.5323	0.31
Bugbear	11.74	30.3636	0.12