03_final_report

December 18, 2025

1 03 Final Report

Data-driven TFT Set 16 meta summary built from cleaned Riot match data.

1.1 Executive Summary

- Winners lean into high-cost boards: avg level **9.22 vs 8.49** and avg unit total cost **7.01 vs 5.66**; they field **3.23 five-cost units** per board vs **2.82** overall while using fewer 1-costs.
- Core trait anchors for winning boards: **Arcanist (1581)**, **Bilgewater (994)**, **Void (870)**, **Freljord (816)**. Non-fast9 winners cluster around **Void/Freljord/Arcanist**, while fast9 flexes keep **Swain/Taric/Wukong** for utility.
- Item edges: most frequent winning items are **Guinsoo's (5054)**, **Thief's Gloves (4329)**, **Adaptive Helm (3233)**; lowest-average-placement items include **Bilgewater Emblem (3.84)**, **Deathblade (3.95)**, **Sterak's (4.00)**.
- High-cost carries dominate: **Shyvana** and **Lucian** top 5-cost presence, often paired with **Juggernaut** and **Gunslinger** traits; placement correlates strongest with **avg_unit_total_cost (0.61)** and **level (0.53)**.

1.2 Introduction

- **Motivation:** Quantify the S16 ranked meta and identify the unit/trait/item patterns that most often appear on top-4 boards.
- **Dataset:** 1,604 matches / 1,310 players (12,757 player-match rows) from processed TFT logs; cleaned via schema + range + reference checks.
- **Methodology:** Validate raw tables (00), explore distributions and correlations (01), then answer targeted meta questions with saved figures/tables (02).
- **Limitations:** No patch-split; assumes reference lookups cover all observed units/traits/items; no MMR control or lobby strength adjustment.

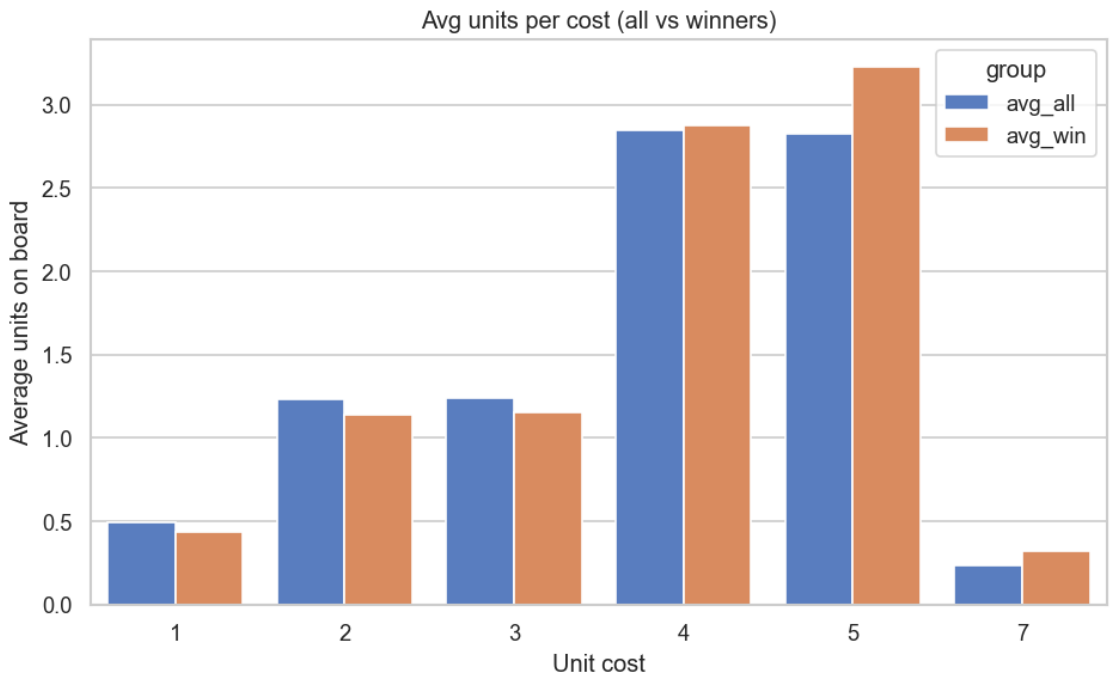
[ ]:
```python
# Setup
from pathlib import Path
import pandas as pd

MARKERS = {"requirements.txt", "Projectplan.md", ".git"}
PROJECT_ROOT = None
cwd = Path.cwd()
for path in [cwd, *cwd.parents]:
    if any((path / m).exists() for m in MARKERS):
        PROJECT_ROOT = path
        break
if PROJECT_ROOT is None:
    PROJECT_ROOT = cwd

DATA_PROCESSED = PROJECT_ROOT / 'data' / 'processed'
CLEANED_DIR = DATA_PROCESSED / 'cleaned'
OUTPUTS_DIR = PROJECT_ROOT / 'outputs'
FIGURES_DIR = OUTPUTS_DIR / 'figures'
TABLES_DIR = OUTPUTS_DIR / 'tables'

participants = pd.read_csv(CLEANED_DIR / 'participants.csv')
traits = pd.read_csv(CLEANED_DIR / 'traits.csv')
units = pd.read_csv(CLEANED_DIR / 'units.csv')
units_ref = pd.read_csv(DATA_PROCESSED / 'canonical_original' / 'units_s16.csv')
unit_cost_map = dict(zip(units_ref['name'], units_ref['cost']))

# Preload analysis tables for quick reference
cost_dist = pd.read_csv(TABLES_DIR / '02_unit_cost_distribution.csv')
unit_pop = pd.read_csv(TABLES_DIR / '02_units_popularity_by_cost.csv')
item_freq = pd.read_csv(TABLES_DIR / '02_items_frequency.csv')
trait_freq = pd.read_csv(TABLES_DIR / '02_traits_frequency_no_exclusive.csv')
winner_traits = pd.read_csv(TABLES_DIR / '02_winner_main_traits.csv')
winner_traits_non_fast9 = pd.read_csv(TABLES_DIR /␣
 ↪'02_winner_main_traits_non_fast9.csv')
fast9_units = pd.read_csv(TABLES_DIR / '02_fast9_non_highcost_units.csv')
high_cost_trait = pd.read_csv(TABLES_DIR / '02_high_cost_trait_cofreq.csv')
high_cost_no_trait = pd.read_csv(TABLES_DIR /␣
 ↪'02_high_cost_units_no_high_traits.csv')

winner_boards = participants[participants['placement'] <=␣
 ↪4][['match_id','puuid']]
WINNER_BOARD_COUNT = len(winner_boards)
```

[ ]:
```
# Dataset overview snapshot
data_overview = {
    'matches': participants['match_id'].nunique(),
    'players': participants['puuid'].nunique(),
    'player_match_rows': len(participants),
    'win_rate_top4': (participants['placement'] <= 4).mean(),
    'avg_level': participants['level'].mean(),
}
pd.DataFrame([data_overview])
```

1.3 Analysis Results

1.3.1 Q1. Cost distribution for winners

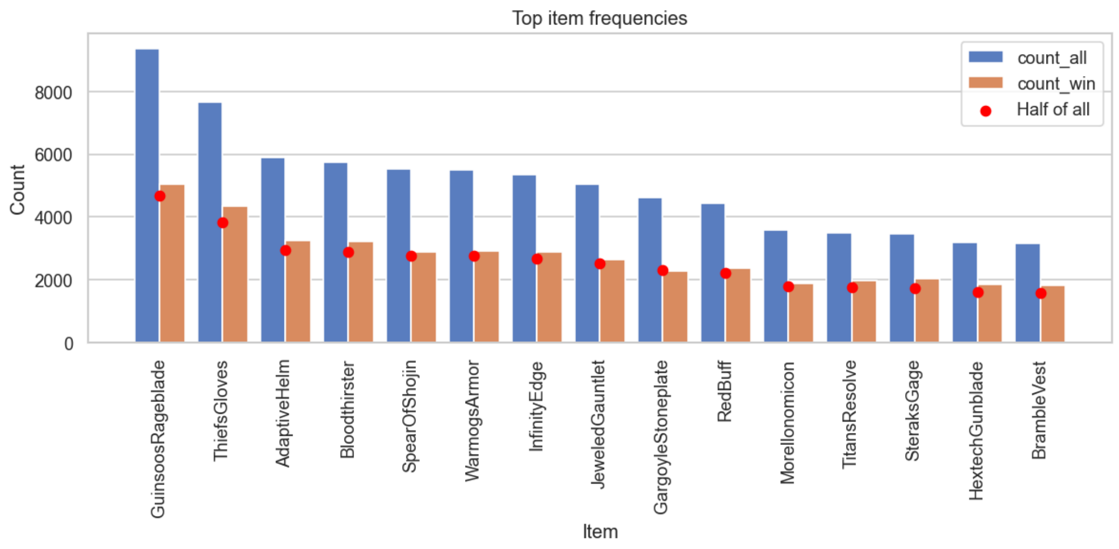
- Winners field **3.23 five-cost** units on average vs **2.82** across all players; 1-cost usage drops from **0.49 to 0.44**.
- Suggests aggressive leveling/econ pays off more than low-cost reroll in this dataset.



```
[ ]: cost_dist
```

1.3.2 Q2. Item popularity and lift

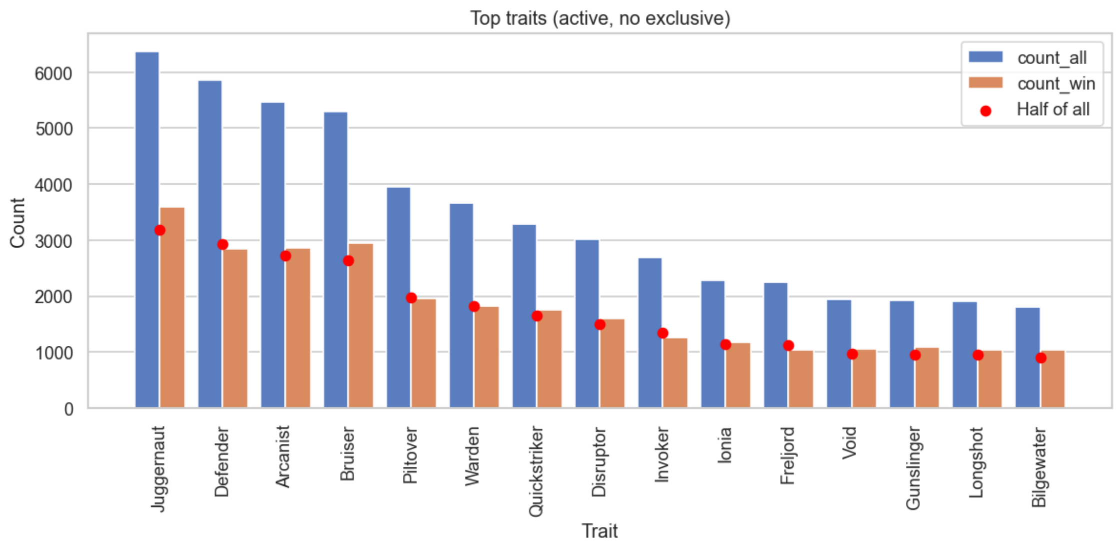
- Most common winning items: **Guinsoo's (5054)**, **Thief's Gloves (4329)**, **Adaptive Helm (3233)**.
- Lowest-average-placement items (from EDA): **Bilgewater Emblem (3.84)**, **Deathblade (3.95)**, **Sterak's (4.00)**.
- Heavy reliance on flexible slam items rather than niche artifacts; emblems appear but are rarer.

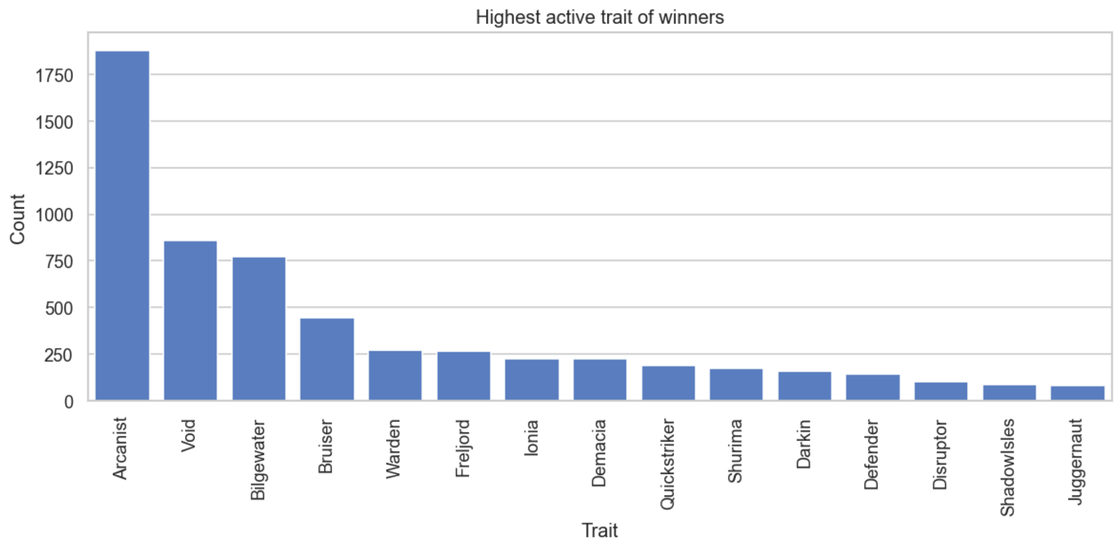


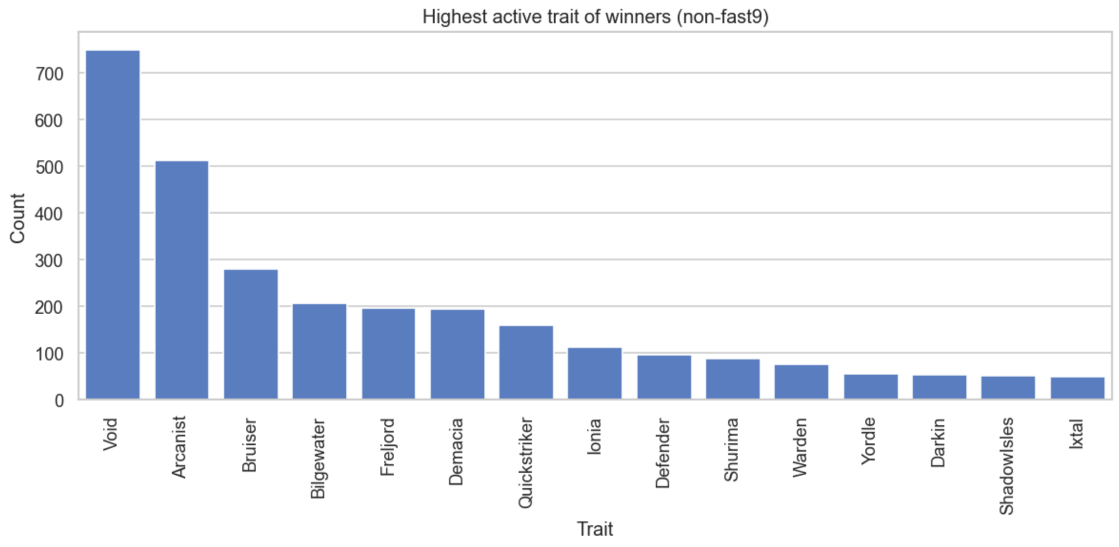
```
[ ]: item_freq.head(15)
```

1.3.3 Q3. Traits popularity and main anchors

- Active non-exclusive trait counts (winners) led by **Juggernaut/Arcanist/Defender/Bruiser**; see plot.
- Highest active trait per winner: **Arcanist 1581**, **Bilgewater 994**, **Void 870**, **Freljord 816**.
- Non-fast9 boards lean **Void/Freljord/Arcanist/Demacia**; fast9 players plug in supportive low-costs like **Swain/Taric/Wukong** while rushing legendaries.



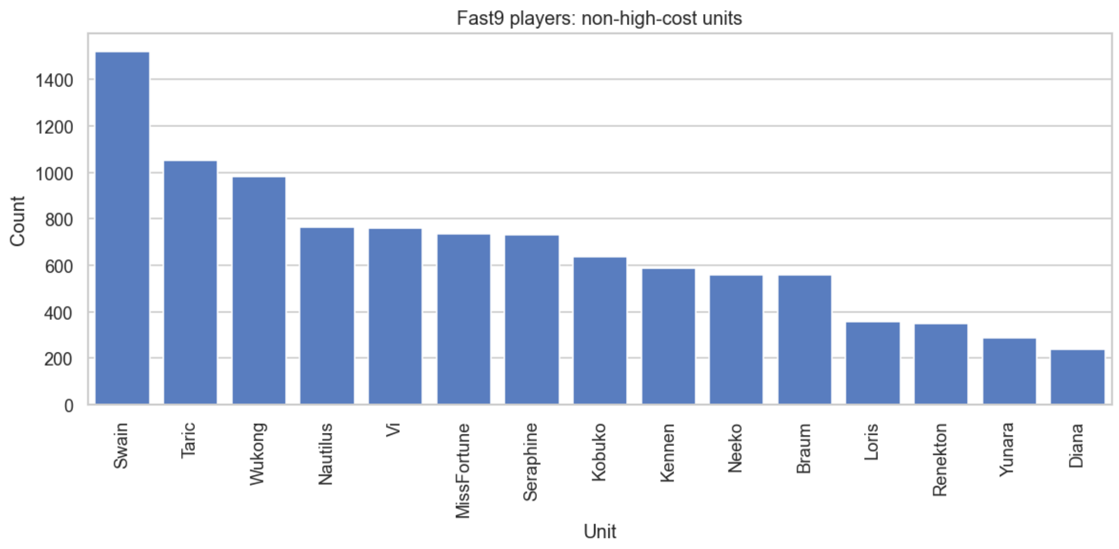




```
[ ]: winner_traits.head(), winner_traits_non_fast9.head()
```

1.3.4 Q4. Popular/powerful non-5/7 cost units

- Fast9 players most often keep **Swain (1522)**, **Taric (1054)**, **Wukong (984)** on boards despite pivoting to 5-costs, implying strong utility/CC value.
- Non-legendary staples can be highlighted for mid-game stabilization.



```
[ ]: fast9_units.head()
```

1.3.5 Q4b. Fully itemized three-star carries/tanks

- Only **5.9%** of winning boards fielded any fully itemized 3-star unit (3 items on a 3-star).
- Top capped units by board rate (share of all top-4 boards): **Tryndamere 1.4%**, **Bard 0.6%**, **Neeko 0.45%**, **Nautilus 0.44%**, **Draven 0.39%**.
- These are mostly low-cost frontline/utility pieces; fully itemized 3-star carries/tanks are rare in winning comps, implying most wins rely on 2-star legendaries instead.

```
[ ]: # Fully itemized three-star carries/tanks on winner boards
item_cols = [c for c in units.columns if c.startswith('item_')]
units_win = units.merge(winner_boards, on=['match_id','puuid'], how='inner')
three_star_full = units_win.dropna(subset=item_cols)
three_star_full = three_star_full[three_star_full['unit_tier'] == 3].copy()
three_star_full['unit_cost'] = three_star_full['unit_name'].map(unit_cost_map)

three_star_summary = (
    three_star_full
    .groupby(['unit_name','unit_cost'])
    .size()
    .reset_index(name='count')
    .sort_values('count', ascending=False)
)
three_star_summary['board_rate'] = three_star_summary['count'] /␣
 ↪WINNER_BOARD_COUNT

boards_with_any_three_star = three_star_full[['match_id','puuid']].
 ↪drop_duplicates()
three_star_presence_rate = len(boards_with_any_three_star) / WINNER_BOARD_COUNT
```

```
print(f'Boards with fully itemized 3-star: {len(boards_with_any_three_star)} /␣
↪{WINNER_BOARD_COUNT} ({three_star_presence_rate:.2%})')
three_star_summary.head(10)
```

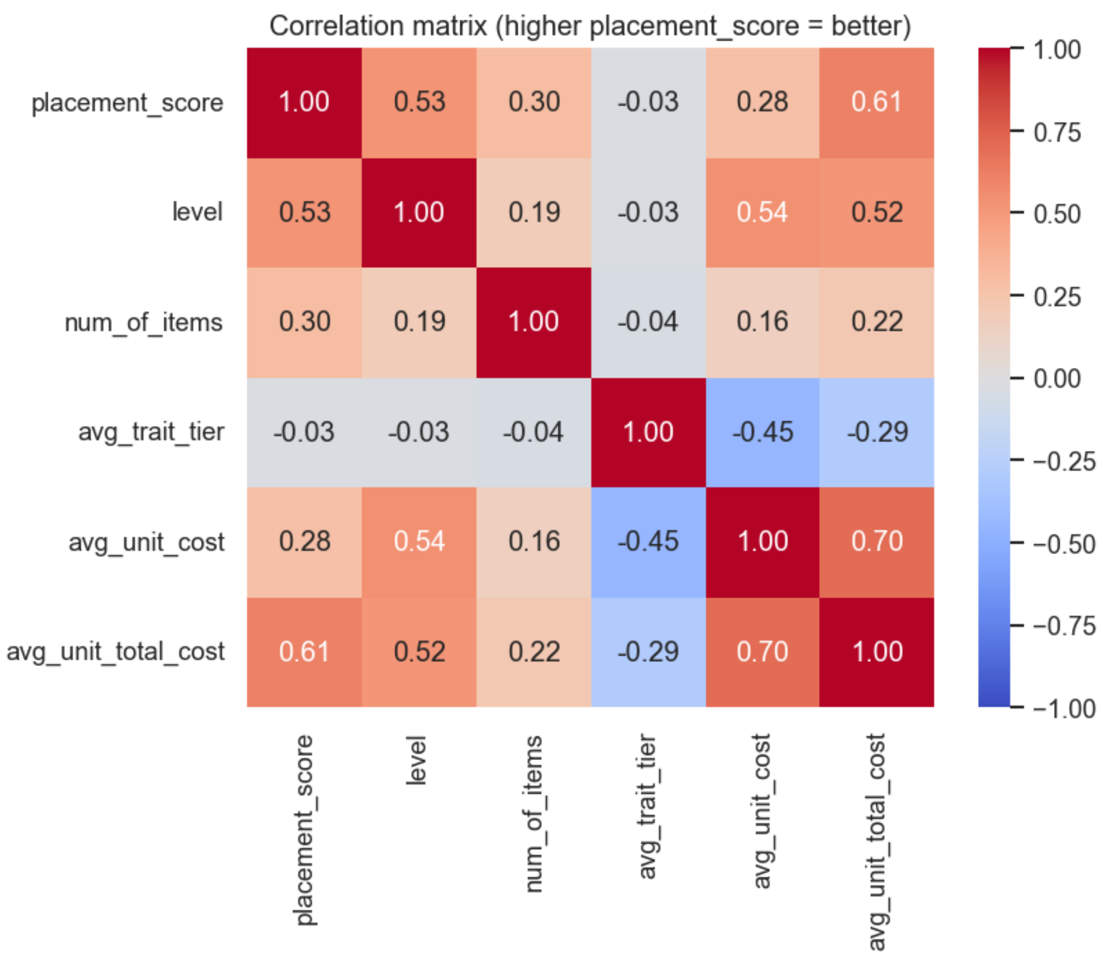
1.3.6 Q5. High-cost units and trait context

- Most frequent high-cost units on winning boards: **Shyvana**, **Lucian**, **Fiddlesticks**, **Azir**, **Kindred**.
- Trait co-occurrence shows **Gunslinger (Lucian 3.2k)** and **Juggernaut (Shyvana 2.8k)** as common caps.
- A subset of legendary boards win without any high-tier non-exclusive traits active (counts in table below).

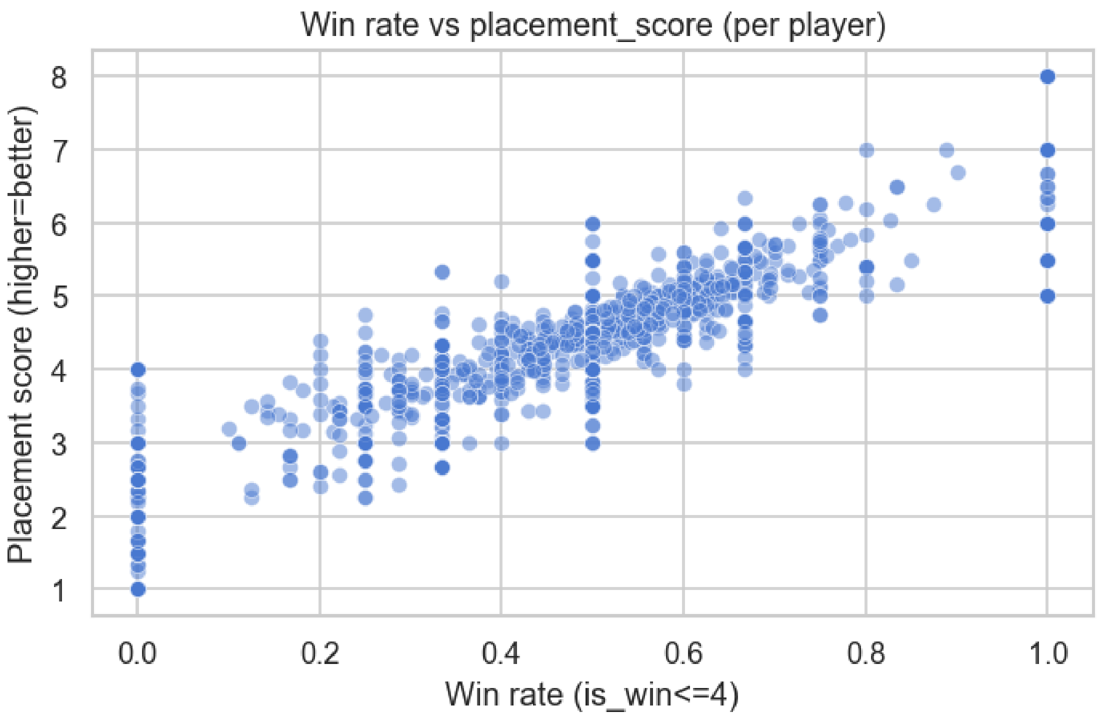
```
[ ]: high_cost_trait.head(10), high_cost_no_trait.sort_values('count',␣
↪ascending=False).head(10)
```

1.3.7 Q6. Winner identifiers (board-level features)

- Correlations vs placement_score: **avg_unit_total_cost 0.61**, **level 0.53**, **avg_unit_cost 0.28**, **num_of_items 0.30**. Trait tier average is weakly negative (likely because high-cost comps run fewer trait caps).
- Winners average **9.22 level**, **~9 items**, **3.93 unit_cost**, **7.01 total unit cost**; others sit at **8.49 level** and **5.66 total unit cost**.



	placement_score	level	num_of_items	avg_trait_tier	avg_unit_cost	avg_unit_total_cost
placement_score	1.00	0.53	0.30	-0.03	0.28	0.61
level	0.53	1.00	0.19	-0.03	0.54	0.52
num_of_items	0.30	0.19	1.00	-0.04	0.16	0.22
avg_trait_tier	-0.03	-0.03	-0.04	1.00	-0.45	-0.29
avg_unit_cost	0.28	0.54	0.16	-0.45	1.00	0.70
avg_unit_total_cost	0.61	0.52	0.22	-0.29	0.70	1.00



```
[ ]: # Recompute board-level summary and correlation for transparency
     item_cols = [c for c in units.columns if c.startswith('item_')]
     items_long = units[['match_id','puuid'] + item_cols].
      ↪set_index(['match_id','puuid']).stack(dropna=True).reset_index()
     items_long.columns = ['match_id','puuid','slot','item']

     participants_scored = participants.assign(placement_score = 9 -␣
      ↪participants['placement'])
     unit_cost_map = dict(zip(pd.read_csv(DATA_PROCESSED / 'canonical_original' /␣
      ↪'units_s16.csv')['name'],
                              pd.read_csv(DATA_PROCESSED / 'canonical_original' /␣
      ↪'units_s16.csv')['cost']))
     units_cost_df = units.copy()
     units_cost_df['unit_cost'] = units_cost_df['unit_name'].map(unit_cost_map)
     units_cost_df['unit_total_cost'] = units_cost_df['unit_cost'] *␣
      ↪units_cost_df['unit_tier']

     unit_counts = units.groupby(['match_id','puuid']).size().
      ↪rename('units_per_board')
     items_per_player = items_long.groupby(['match_id','puuid']).size().
      ↪rename('num_of_items')
     unit_cost_avg = units_cost_df.groupby(['match_id','puuid'])['unit_cost'].mean().
      ↪rename('avg_unit_cost')
```

```
unit_total_cost_avg = units_cost_df.
 ↪groupby(['match_id','puuid'])['unit_total_cost'].mean().
 ↪rename('avg_unit_total_cost')
avg_trait_tier = traits.groupby(['match_id','puuid'])['tier_current'].mean().
 ↪rename('avg_trait_tier')

player_df = (participants_scored
     .merge(unit_counts, on=['match_id','puuid'], how='left')
     .merge(items_per_player, on=['match_id','puuid'], how='left')
     .merge(unit_cost_avg, on=['match_id','puuid'], how='left')
     .merge(unit_total_cost_avg, on=['match_id','puuid'], how='left')
     .merge(avg_trait_tier, on=['match_id','puuid'], how='left')
)

corr_cols =␣
 ↪['placement_score','level','num_of_items','avg_trait_tier','avg_unit_cost','avg_unit_total_
correlations = player_df[corr_cols].corr()
summary = player_df.assign(is_win_flag = player_df['placement'] <= 4).
 ↪groupby('is_win_flag').agg({
    'placement':'mean',
    'level':'mean',
    'num_of_items':'mean',
    'avg_trait_tier':'mean',
    'avg_unit_cost':'mean',
    'avg_unit_total_cost':'mean'
}).rename(index={True:'winners', False:'others'})
summary, correlations
```

1.4 Conclusions

- **High-cost focus wins:** Rushing 9 and stuffing boards with multiple 5-costs is the clearest differentiator; low-cost reroll is underperforming in this sample.
- **Trait anchors vary by econ path:** Fast9 boards anchor on Arcanist/Bilgewater while non-fast9 lean Void/Freljord/Demacia.
- **Items reward tempo:** Universal slam items (Guinsoo's, TG, Adaptive Helm) dominate presence; Bilgewater Emblem/Deathblade/Sterak's show best average placements.
- **Future work:** Patch-diff splits, matchup-adjusted performance, and comp-level clustering to surface full templates; add external meta site comparison (Phase 6).

1.5 Appendix

- Figures live in `outputs/figures/`; tables in `outputs/tables/`.
- Re-run upstream notebooks (00, 01, 02) to refresh after new data or patches.
- Export HTML via `jupyter nbconvert --to html notebooks/03_final_report.ipynb --output ../outputs/report.html`.