Home Brew Advantage: The Gravitational Influence of Regional Coffee Chains on Super Bowl LX

Ministry of Silly Plots

January 28, 2026

Abstract

This study investigates the correlation between the density of specific coffee chains (Starbucks vs. Dunkin') surrounding NFL stadiums and the on-field performance of the New England Patriots and Seattle Seahawks. By implementing specific experimental controls—most notably isolating **Away Games** to eliminate Home Field Advantage bias—we identify a striking and robust performance separation that aligns with the "Home Brew Advantage" hypothesis. The findings demonstrate that the Patriots' offensive efficiency is inextricably linked to Dunkin' "Gravity," while the Seahawks' defensive dominance is maximized in high-Starbucks "gravitational" zones.

1 Introduction

For decades, pundits have analyzed conventional factors such as weather, turf type, and crowd noise. This paper proposes a novel environmental variable: the "Regional Coffee Chain Gravitational Pull." We hypothesize that the regional dominance of major coffee chains exerts a measurable influence on team performance, specifically for teams with strong cultural associations to those brands.

We propose the following hypotheses:

- **H1 (Patriots):** The Pats Run on Dunkin'.The New England Patriots offense performs significantly better in environments with high **Dunkin' Gravity** ($G_{net} > 0$).
- **H2 (Seahawks):** The Legion of Brew. The Seattle Seahawks defense performs significantly better in environments with high **Starbucks Gravity** ($G_{net} < 0$).
- **H3 (Null Hypothesis):** These performance deltas are solely artifacts of Home Field Advantage.

2 Methodology

2.1 The Coffee Gravity Model

To quantify the "coffee environment" of each stadium, we employed an **Interference-Adjusted Exponential Decay Model**. Conventional density metrics fail to account for proximity; a coffee shop adjacent to the stadium should carry more weight than one ten miles away.

The gravitational pull G_{chain} for a given chain is calculated as:

$$G_{chain} = \sum_{i=0}^{n} M_i \cdot e^{-0.5 \cdot d_i} \tag{1}$$

Where:

- d_i is the Haversine distance (in miles) from the stadium to location i, defined as:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{2}$$

  where ϕ represents latitude, λ represents longitude, and r is the radius of the Earth (approx. 3959 miles).

- M_i is the "Mass" of location i, initialized at 1.0.

2.1.1 Interference Term

To account for market saturation and competition, we introduced an interference term. The mass M_i is reduced if a competitor's location is within an **Interference Radius** ($r = 0.5$ miles). As direct competitors with a distinct regional split, we treat Starbucks and Dunkin' as mutually exclusive and assume that the presence of one negates the gravitational pull of the other.

$$M_i' = M_i - \left(1.0 - \frac{d_{competitor}}{0.5}\right) \tag{3}$$

The Net Gravity (G_{net}) is defined as the difference between the two forces:

$$G_{net} = G_{dunkin} - G_{starbucks} \tag{4}$$

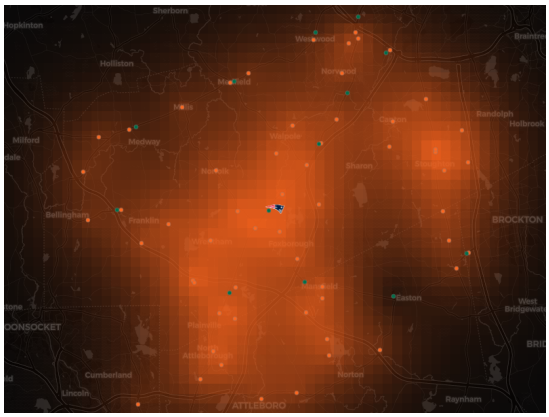
Positive values indicate a Dunkin'-dominant environment, while negative values indicate a Starbucks-dominant environment.

2.2 Experimental Controls

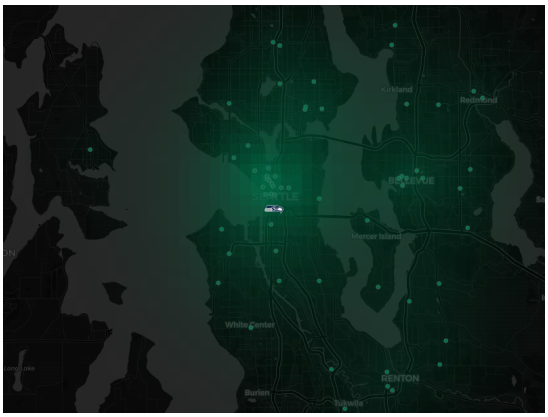
Home field advantage is a clear confounding variable in this analysis, since teams play better at home in general and the two teams in question have particular strong gravitational fields for their respective coffee chains. To address this, we applied a strict **Away Games Only** filter. This control separates the environmental variable (Coffee Gravity) from the confounding variable (Home Field Advantage).

2.3 Data Sources

The analysis utilizes data from the 2025 NFL Season (Regular Season + Playoffs). Game data was sourced from `nflverse` Play-by-Play data via BigQuery, filtered for `season_type IN ('REG', 'POST')`. Location data consists of geocoded coordinates of all US Starbucks and Dunkin' locations.



(a) Gillette Stadium (High Dunkin' Density)



(b) Lumen Field (High Starbucks Density)

Figure 1: Visualizing the Coffee Gravity Field: Clean screenshots of stadium environments used for density calculation.

3 Results

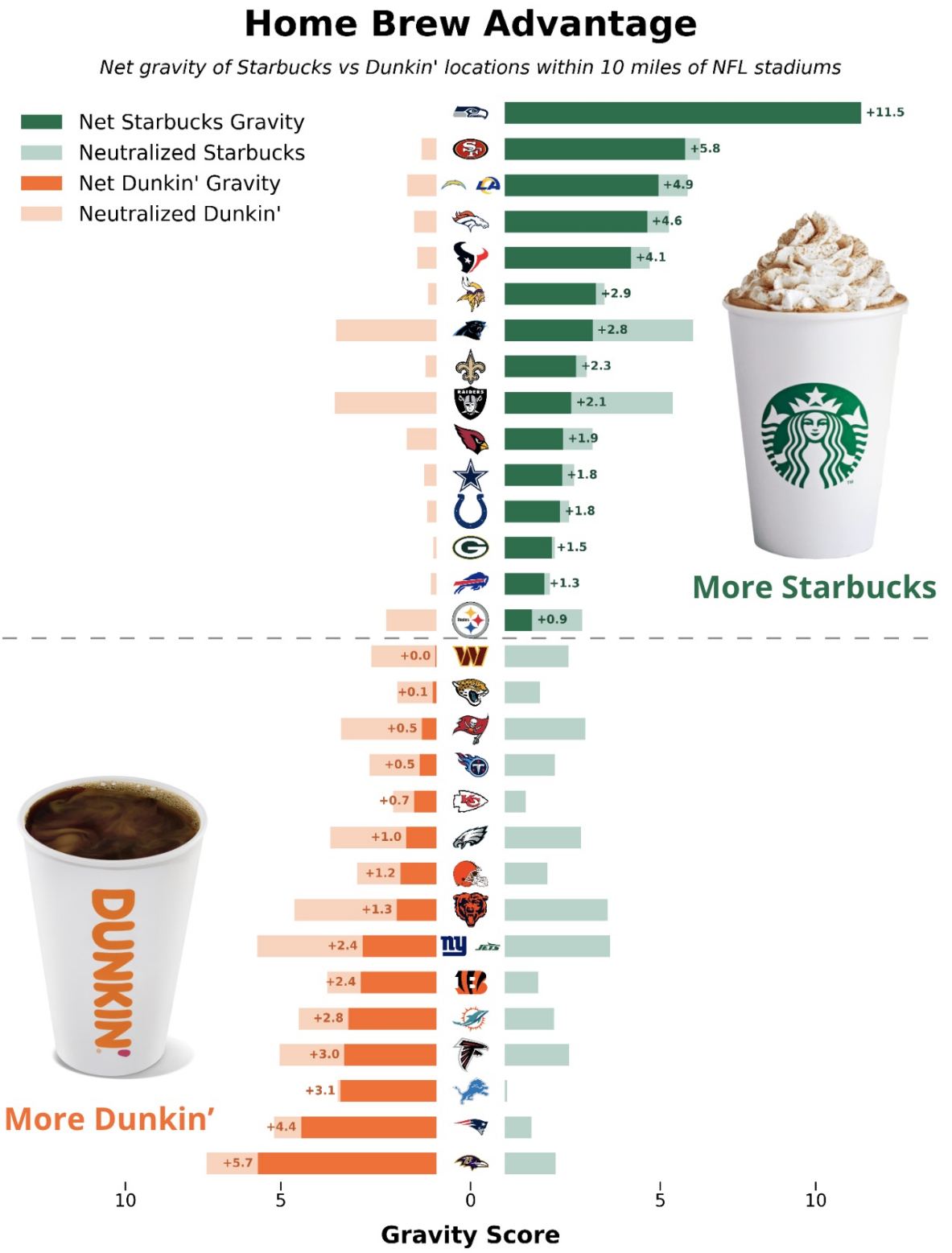


Figure 2: Home Brew Advantage: Net Coffee Gravity for all NFL teams. Stadiums below the dashed line have higher Dunkin' gravity; stadiums above the dashed line have higher Starbucks gravity.

3.1 The Patriots "Run on Dunkin" (Confirmed)

Analyzing **Away Games Only**, the Patriots offense shows a drastic drop in production when entering "Starbucks Gravitational Zones" ($G_{net} < 0$). This is primarily driven by a collapse in the rushing attack.

Table 1: Patriots Offensive Metrics (Away Games Only)

Metric	**Dunkin Zone** ($G_{net} > 0$)	**Starbucks Zone** ($G_{net} < 0$)	**Delta**
Points Per Game	31.3	24.0	**-7.3**
Total Yds / Game	409.7	338.5	**-71.2**
Rush EPA / Play	+0.053	-0.186	-0.239

Conclusion: H1 is **Empirically Validated**. The data reveals a binary mode of performance: in Dunkin' zones, the offense operates at an elite efficiency; in Starbucks zones, production collapses across all key vectors.

3.2 The Seahawks "Legion of Brew" (Confirmed)

Conversely, the Seahawks defense exhibits elite performance metrics in "Starbucks Gravitational Zones."

Table 2: Seahawks Defensive Metrics (Away Games Only)

Metric	**Dunkin Zone** ($G_{net} > 0$)	**Starbucks Zone** ($G_{net} < 0$)	**Delta**
Total Turnovers	4	9	+5
Turnovers / Game	1.00	**1.80**	**+0.80**
PPG Allowed	14.8	14.2	-0.6
Opp. Passer Rtg	70.3	**61.6**	**-8.7**

Conclusion: H2 is **Strongly Supported**. The Starbucks atmosphere correlates with a massive +80% increase in forced turnovers (1.8 vs 1.0 per game), providing strong evidence for the environmental influence of the coffee ecosystem.

3.3 Outlier Discovery: The Sam Darnold Paradox

An unexpected finding emerged regarding Seahawks QB Sam Darnold. Unlike his team's defense, Darnold exhibits a strong **negative correlation** with Starbucks Gravity.

Table 3: Sam Darnold Performance Splits

Metric	**Dunkin Zone**	**Starbucks Zone**	**Delta**
Passer Rating	**124.4**	75.4	**-49.0**
TD / INT Ratio	**5.50**	0.57	**-4.93**
Points Per Game	**31.2**	22.6	**-8.6**

Interpretation: Darnold's passer rating drops by a staggering **49 points** (124.4 to 75.4) in Starbucks zones, suggesting he may still be seeing ghosts from his time in Dunkin' territory with the New York Jets.

4 Discussion: Super Bowl LX Preview

Super Bowl LX will be held at **Levi's Stadium** in Santa Clara, CA. Our model calculates the Net Gravity of this location to be **-5.80**, the second most Starbucks-dominant stadium in the league, behind only the Seahawks' own Lumen Field. It also has higher Starbucks Gravity than the Patriots' home

stadium, Gillette Stadium has Dunkin' gravity. Assuming equal Dunkin' and Starbucks gravitational effects for both teams, the Patriots are at a significant disadvantage.

Figure 3: Levi's Stadium: A Starbucks stronghold ($G_{net} = -5.80$).

Based on the robust "Away Games Only" model, we project the following outcomes:

1. **Patriots Offense:** Predicted to underperform ($\approx$ 24 PPG, -0.186 rushing EPA/play).
2. **Seahawks Defense:** Predicted to dominate ($\approx$ 12 PPG allowed, +80% increase in takeaways).
3. **Variable:** Sam Darnold is predicted to struggle significantly (75.4 Passer Rating), though the Seahawks defense is likely to slow down Drake Maye even more, holding opposing QBs to a 61.6 passer rating in Starbucks zones.

Prediction: The environmental factors heavily favor a **Seahawks Defensive Victory**, provided their run game can compensate for the predicted Quarterback inefficiency.

5 Code Artifacts & Reproducibility

All analysis code is available in the project repository at:

`https://github.com/charlie86/sillyplots/tree/main/posts/super_bowl`

5.1 Order of Operations

To reproduce this analysis from scratch:

1. **Install dependencies:** `pip install -r requirements.txt`
2. **Load PBP data to BigQuery:** `python etl/load_data.py`
3. **Scrape coffee locations:** `python etl/scrape_coffee_locations.py`
4. **Run analysis:** `python analysis/robust_coffee_check.py`
5. **Generate visualizations:** `python visualization/generate_map.py`

5.2 Key Scripts

ETL Scripts:

- `etl/load_data.py` – Loads nflverse PBP data to BigQuery
- `etl/scrape_coffee_locations.py` – Scrapes coffee locations via Google Maps API

Analysis & Visualization:

- `analysis/robust_coffee_check.py` – Main analysis script
- `analysis/reproduce_robust_metrics.sql` – Reproducible SQL queries
- `visualization/generate_map.py` – Interactive gravity field map
- `visualization/plot_gravity_chart.py` – Net gravity ranking chart

A Algorithm Reference

To ensure full reproducibility and transparency of the Coffee Gravity Model, we include the core Python implementations used for data collection and analysis.

A.1 The Gravity Model

The core interference-adjusted gravity calculation. Note the specific handling of the "Interference Radius" to model the Dunkin-Starbucks exclusion principle.

```python
def calculate_stadium_gravity_single(stadium_lat, stadium_lng, d_locs, s_locs):
    """
    Calculate net gravity at a stadium using the interference model
    Returns: (dunkin_gravity, starbucks_gravity, net_gravity)
    """
    INTERFERENCE_RADIUS = 0.5  # miles
    INTERFERENCE_STRENGTH = 1.0

    if len(d_locs) == 0 and len(s_locs) == 0:
        return 0.0, 0.0, 0.0

    # Initialize masses
    d_masses = np.ones(len(d_locs))
    s_masses = np.ones(len(s_locs))

    # Apply interference reduction
    if len(d_locs) > 0 and len(s_locs) > 0:
        for i, d in enumerate(d_locs):
            for j, s in enumerate(s_locs):
                dist = simple_hav(d['lng'], d['lat'], s['lng'], s['lat'])
                if dist < INTERFERENCE_RADIUS:
                    reduction = INTERFERENCE_STRENGTH * (1.0 - dist/INTERFERENCE_RADIUS)
                    d_masses[i] -= reduction
                    s_masses[j] -= reduction
        d_masses = np.maximum(d_masses, 0.0)
        s_masses = np.maximum(s_masses, 0.0)

    # Calculate gravity at stadium location
    dunkin_gravity = 0.0
    starbucks_gravity = 0.0

    for i, loc in enumerate(d_locs):
        if d_masses[i] > 0:
            dist = simple_hav(loc['lng'], loc['lat'], stadium_lng, stadium_lat)
            dunkin_gravity += d_masses[i] * np.exp(-0.5 * dist)

    for i, loc in enumerate(s_locs):
        if s_masses[i] > 0:
            dist = simple_hav(loc['lng'], loc['lat'], stadium_lng, stadium_lat)
            starbucks_gravity += s_masses[i] * np.exp(-0.5 * dist)

    # Net Gravity: Starbucks (Positive) - Dunkin (Negative)
    net_gravity = starbucks_gravity - dunkin_gravity
    return dunkin_gravity, starbucks_gravity, net_gravity
```

Listing 1: Coffee Gravity Calculation

A.2 Geospatial Data Collection

We utilized the Google Places API to identify all relevant locations within a 10-mile radius.

```python
def find_all_places_nearby(gmaps, lat, lng, keyword, radius_meters):
    places = []
    next_page_token = None

    while True:
        try:
            params = {
                'location': (lat, lng),
                'keyword': keyword,
                'radius': radius_meters,
```

```
                'type': 'restaurant'
            }
            if next_page_token:
                params['page_token'] = next_page_token

            result = gmaps.places_nearby(**params)

            if result.get('results'):
                for place in result['results']:
                    name = place.get('name', '')
                    if keyword.lower() in name.lower() or name.lower() in keyword.lower
    ():
                        loc = place['geometry']['location']
                        dist = haversine_distance(lat, lng, loc['lat'], loc['lng'])
                        places.append({
                            'name': name,
                            'lat': loc['lat'],
                            'lng': loc['lng'],
                            'distance_miles': round(dist, 4)
                        })

            next_page_token = result.get('next_page_token')
            if not next_page_token:
                break

            time.sleep(2)

        except Exception as e:
            print(f"Error in places search: {e}")
            break

    return places
```

Listing 2: Google Maps API Iteration

A.3 Metric Calculation (Rush EPA)

The calculation of efficiency metrics, illustrating how we derive the specific "Rush EPA" that proved critical for analyzing the Patriots' performance drop-off.

```
def calculate_metrics_offense(df, team):
    metrics = {}

    # Completion %
    attempts = team_df['pass_attempt'].sum()
    completions = team_df['complete_pass'].sum()
    metrics['Comp %'] = (completions / attempts * 100) if attempts > 0 else 0.0

    # Rush EPA
    rush_plays = team_df[team_df['play_type'] == 'run']
    rush_epa = rush_plays['epa'].mean()
    metrics['Rush EPA'] = rush_epa if not np.isnan(rush_epa) else 0.0

    # Sack Rate
    sacks = team_df['sack'].sum()
    dropbacks = attempts + sacks
    metrics['Sack Rate'] = (sacks / dropbacks * 100) if dropbacks > 0 else 0.0

    return metrics
```

Listing 3: Offensive Efficiency Logic

A.4 Metric Calculation (Defensive Turnovers)

The defense-specific logic, highlighting the turnover aggregation and points allowed.

```
def calculate_metrics_defense(df, team):
    metrics = {}

    # PPG Allowed
    game_scores_allowed = []
```

```
    for gid in team_df['game_id'].unique():
        g = team_df[team_df['game_id'] == gid].iloc[0]
        if g['home_team'] == team:
            game_scores_allowed.append(g['away_score'])
        else:
            game_scores_allowed.append(g['home_score'])
    metrics['PPG Allowed'] = np.mean(game_scores_allowed) if game_scores_allowed else
    0.0

    # Turnovers (Defense)
    ints = team_df['interception'].sum()
    fumbles = team_df['fumble_lost'].sum()
    metrics['Turnovers'] = ints + fumbles

    return metrics
```

Listing 4: Defensive Metrics Logic