# Pythogorean expectation

## Definition

Pythagorean winning percentage is a formula developed by renowned statistician Bill James. The concept strives to determine the number of games that a team *should* have won -- based its total number of runs scored versus its number of runs allowed -- in an effort to better forecast that team's future outlook.

The initial formula for pythagorean winning percentage was as follows: (runs scored ^ 2) / [(runs scored ^ 2) + (runs allowed ^ 2)] That formula proved more predictive than basic winning percentage when trying to predict a team's future performance, although in the years since pythagorean winning percentage was popularized, other analysts have attempted to find an even more accurate formula.

Baseball-Reference.com (https://www.baseball-reference.com/), for instance, uses 1.83 as its exponent of choice -- a modification that has successfully narrowed the formula's margin of error.

## Why it's useful ¶

Pythagorean winning percentage can help to identify teams that have either overachieved or underachieved. When looking at a club with a surprisingly poor or surprisingly strong record early in the season, using the theory to determine a team's "expected" winning percentage for the remainder of the year can paint a more accurate picture of how things will play out than merely looking at actual winning percentage.

## Codigo

### Importar bibliotecas y conexion a la base de datos

In [1]:
```
from sqlalchemy import create_engine, text
import dotenv
import os
import pandas as pd
import numpy as np
import datetime
import statsmodels.formula.api as smf
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go
import plotly.io as pio
pio.renderers.default = 'notebook'
```

In [2]:
```
def crearEngine(coneccion_local=True):
    dotenv.load_dotenv()
    if coneccion_local:
        user = os.getenv('DB_USER_LOCAL')
        password = os.getenv('DB_PASSWORD_LOCAL')
        host = os.getenv('DB_HOST_LOCAL')
        port = os.getenv('DB_PORT_LOCAL')
        database = os.getenv('DB_NAME_LOCAL')
        connection_url = f'postgresql+psycopg2://{user}:{password}@{host}:{port}/{database}'
    else:
        user = os.getenv('DB_USER_PROD')
        password = os.getenv('DB_PASSWORD_PROD')
        host = os.getenv('DB_HOST_PROD')
        port = os.getenv('DB_PORT_PROD')
        database = os.getenv('DB_NAME_PROD')
        connection_url=f'postgresql://{user}:{password}@{host}/{database}?sslmode=require'

    return create_engine(connection_url)
engine = crearEngine()
```

## Validar si esta relacion se aplica en baseball

Vamos a obtener los datos de todas las temporadas para calcular el pythagorean expectation de cada equipo y porcentaje de victorias real al final de cada temporada.

Obtener y preparar los datos

In [3]:
```
query = text("""select extract(year from g.primer_lanzamiento)::integer as temporada, loc.nombre as home_team,
              vis.nombre as visiting_team, g.carreras_local as home_r, g.carreras_visitante as vis_r
              from juego g
              join equipo loc on g.local_id = loc.equipo_id
              join equipo vis on g.visitante_id = vis.equipo_id""")
lmb = pd.read_sql(query, engine)

# se agregan estas columnas para poder aplicar funciones de agregación
lmb['hwin'] = np.where(lmb['home_r'] > lmb['vis_r'], 1, 0)
lmb['awin'] = np.where(lmb['home_r'] < lmb['vis_r'], 1, 0)
lmb['count'] = 1
```

Separar en 2 df uno para los partidos de local y otro para los de visitante

In [4]:
```
lmbTemporadaLocal = lmb.groupby(['temporada', 'home_team'])[['hwin', 'home_r', 'vis_r', 'count']].sum().reset_index()
lmbTemporadaLocal = lmbTemporadaLocal.rename(columns={'home_team': 'team', 'home_r': 'home_r_h', 'vis_r': 'vis_r_h', 'count': 'G_h'})
```

In [5]:
```
lmbTemporadaAway = lmb.groupby(['temporada', 'visiting_team'])[['awin', 'home_r', 'vis_r', 'count']].sum().reset_index()
lmbTemporadaAway = lmbTemporadaAway.rename(columns={'visiting_team': 'team', 'home_r': 'home_r_a', 'vis_r': 'vis_r_a', 'count': 'G_a'})
```

Unir ambos df para obtener toda la informacion de la temporada

In [6]:
```
lmbTemporada = pd.merge(lmbTemporadaLocal, lmbTemporadaAway, on=['temporada', 'team'])
```

Agregar nuevas columnas: vitorias totales, juegos totales, carreras anotadas, carreras en contra y porcentaje de victorias

In [7]:
```
lmbTemporada['W_season'] = lmbTemporada['hwin'] + lmbTemporada['awin']
lmbTemporada['G_season'] = lmbTemporada['G_h'] + lmbTemporada['G_a']
lmbTemporada['R_season'] = lmbTemporada['home_r_h'] + lmbTemporada['vis_r_a']
lmbTemporada['RA_season'] = lmbTemporada['vis_r_h'] + lmbTemporada['home_r_a']
lmbTemporada['wpct_season'] = round(lmbTemporada['W_season'] / lmbTemporada['G_season'], 3)
lmbTemporada['wpct_home'] = round(lmbTemporada['hwin'] / lmbTemporada['G_h'], 3)
lmbTemporada['wpct_away'] = round(lmbTemporada['awin'] / lmbTemporada['G_a'], 3)
```

Calcular pythagorean winning percentage

In [8]:
```
lmbTemporada['pyth_season'] = round((lmbTemporada['R_season'] ** 2) / (lmbTemporada['R_season'] ** 2 + lmbTem
porada['RA_season'] ** 2), 3)
lmbTemporada['pyth_home'] = round((lmbTemporada['home_r_h'] ** 2) / (lmbTemporada['home_r_h'] ** 2 + lmbTempo
rada['vis_r_h'] ** 2), 3)
lmbTemporada['pyth_away'] = round((lmbTemporada['vis_r_a'] ** 2) / (lmbTemporada['vis_r_a'] ** 2 + lmbTempora
da['home_r_a'] ** 2), 3)
lmbTemporada.head(5)
```

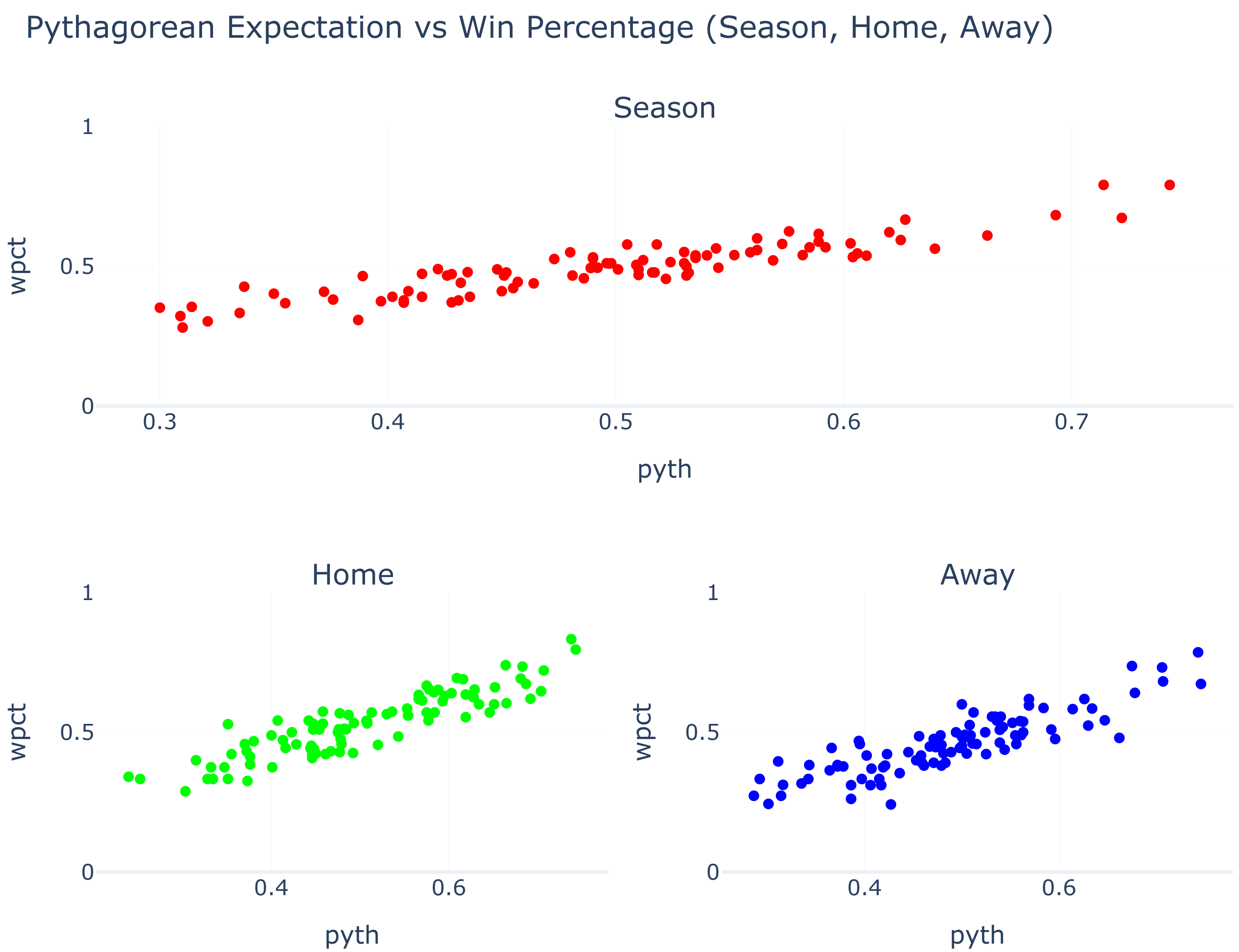
Out[8]:

| | temporada | team | hwin | home_r_h | vis_r_h | G_h | awin | home_r_a | vis_r_a | G_a | W_season | G_season | R_season | RA_season |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2021 | Acereros del Norte | 26 | 213 | 183 | 39 | 14 | 236 | 184 | 37 | 40 | 76 | 397 | 419 |
| **1** | 2021 | Algodoneros Union Laguna | 17 | 182 | 217 | 36 | 17 | 214 | 196 | 35 | 34 | 71 | 378 | 431 |
| **2** | 2021 | Bravos de Leon | 14 | 198 | 219 | 33 | 15 | 194 | 186 | 33 | 29 | 66 | 384 | 413 |
| **3** | 2021 | Caliente de Durango | 11 | 186 | 253 | 33 | 9 | 223 | 141 | 33 | 20 | 66 | 327 | 476 |
| **4** | 2021 | Diablos Rojos del Mexico | 20 | 250 | 259 | 39 | 30 | 135 | 209 | 41 | 50 | 80 | 459 | 394 |

Graficar para comparar el porcentaje de victorias real con el pythagorean winning percentage

In [9]:
```
fig = make_subplots(
    rows=2, cols=2,
    specs=[[{"colspan": 2}, None],
           [{}, {}]],
    subplot_titles=("Season","Home","Away"))

fig.add_trace(go.Scatter(
    x=lmbTemporada['pyth_season'],
    y=lmbTemporada['wpct_season'],
    mode='markers',
    marker=dict(color='rgba(255, 0, 0, 1)'),
    hovertext=lmbTemporada[['team', 'temporada', 'R_season', 'RA_season', 'wpct_season', 'pyth_season']]
                        .apply(lambda x: f"""Team: {x['team']}<br>Temporada: {x['temporada']}<br>R:
                        {x['R_season']}<br>RA: {x['RA_season']}<br>WPct: {x['wpct_season']}<br>Pyth: {x
['pyth_season']}""", axis=1),
    hoverinfo='text'
    ), row=1, col=1)

fig.add_trace(go.Scatter(
    x=lmbTemporada['pyth_home'],
    y=lmbTemporada['wpct_home'],
    mode='markers',
    marker=dict(color='rgba(0, 255, 0, 1)'),
    hovertext=lmbTemporada[['team', 'temporada', 'home_r_h', 'vis_r_h', 'wpct_home', 'pyth_home']]
                        .apply(lambda x: f"""Team: {x['team']}<br>Temporada: {x['temporada']}<br>Team R:
                        {x['home_r_h']}<br>Away R: {x['vis_r_h']}<br>WPct: {x['wpct_home']}<br>Pyth: {x
['pyth_home']}""", axis=1),
    hoverinfo='text'
    ), row=2, col=1)

fig.add_trace(go.Scatter(
    x=lmbTemporada['pyth_away'],
    y=lmbTemporada['wpct_away'],
    mode='markers',
    marker=dict(color='rgba(0, 0, 255, 1)'),
    hovertext=lmbTemporada[['team', 'temporada', 'vis_r_a', 'home_r_a', 'wpct_away', 'pyth_away']]
                        .apply(lambda x: f"""Team: {x['team']}<br>Temporada: {x['temporada']}<br>Local R:
                        {x['home_r_a']}<br>Team R: {x['vis_r_a']}<br>WPct: {x['wpct_away']}<br>Pyth: {x
['pyth_away']}""", axis=1),
    hoverinfo='text'
    ), row=2, col=2)
```

```
fig.update_xaxes(title_text='pyth', row=1, col=1)
fig.update_xaxes(title_text='pyth', row=2, col=1)
fig.update_xaxes(title_text='pyth', row=2, col=2)
fig.update_yaxes(title_text='wpct', range=[0, 1], row=1, col=1)
fig.update_yaxes(title_text='wpct', range=[0, 1], row=2, col=1)
fig.update_yaxes(title_text='wpct', range=[0, 1], row=2, col=2)

# Actualizar el layout
fig.update_layout(
    title='Pythagorean Expectation vs Win Percentage (Season, Home, Away)',
    width=800,
    height=600,
    showlegend=False,
    template='plotly_white'
)

fig.show()
```




En la gráfica se obseva que existe una relacion directamente proporcional entre el porcentaje de victorias real y el pythagorean winning percentage, lo que indica que los equipos que anotan más carreras tienden a ganar más partidos.

Crear modelo de regresión lineal

In [10]:
```python
pyth_lm = smf.ols(formula = 'wpct_season ~ pyth_season', data=lmbTemporada).fit()
pyth_lm.summary()
```

Out[10]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | wpct_season | **R-squared:** | 0.833 |
| **Model:** | OLS | **Adj. R-squared:** | 0.831 |
| **Method:** | Least Squares | **F-statistic:** | 457.5 |
| **Date:** | Tue, 10 Jun 2025 | **Prob (F-statistic):** | 1.80e-37 |
| **Time:** | 20:58:21 | **Log-Likelihood:** | 171.56 |
| **No. Observations:** | 94 | **AIC:** | -339.1 |
| **Df Residuals:** | 92 | **BIC:** | -334.0 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 0.0439 | 0.021 | 2.045 | 0.044 | 0.001 | 0.087 |
| **pyth_season** | 0.9082 | 0.042 | 21.388 | 0.000 | 0.824 | 0.993 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 2.491 | **Durbin-Watson:** | 1.862 |
| **Prob(Omnibus):** | 0.288 | **Jarque-Bera (JB):** | 2.064 |
| **Skew:** | 0.232 | **Prob(JB):** | 0.356 |
| **Kurtosis:** | 2.442 | **Cond. No.** | 13.0 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [11]:
```
pyth_lm = smf.ols(formula = 'wpct_home ~ pyth_home', data=lmbTemporada).fit()
pyth_lm.summary()
```

Out[11]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | wpct_home | **R-squared:** | 0.803 |
| **Model:** | OLS | **Adj. R-squared:** | 0.801 |
| **Method:** | Least Squares | **F-statistic:** | 375.5 |
| **Date:** | Tue, 10 Jun 2025 | **Prob (F-statistic):** | 3.08e-34 |
| **Time:** | 20:58:21 | **Log-Likelihood:** | 148.21 |
| **No. Observations:** | 94 | **AIC:** | -292.4 |
| **Df Residuals:** | 92 | **BIC:** | -287.3 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | **coef** | **std err** | **t** | **P>\|t\|** | **[0.025** | **0.975]** |
|---|---|---|---|---|---|---|
| **Intercept** | 0.0848 | 0.024 | 3.601 | 0.001 | 0.038 | 0.132 |
| **pyth_home** | 0.8842 | 0.046 | 19.378 | 0.000 | 0.794 | 0.975 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 2.513 | **Durbin-Watson:** | 1.771 |
| **Prob(Omnibus):** | 0.285 | **Jarque-Bera (JB):** | 1.624 |
| **Skew:** | -0.004 | **Prob(JB):** | 0.444 |
| **Kurtosis:** | 2.356 | **Cond. No.** | 11.0 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [12]:
```
pyth_lm = smf.ols(formula = 'wpct_away ~ pyth_away', data=lmbTemporada).fit()
pyth_lm.summary()
```

Out[12]:

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | wpct_away | **R-squared:** | 0.752 |
| **Model:** | OLS | **Adj. R-squared:** | 0.749 |
| **Method:** | Least Squares | **F-statistic:** | 278.3 |
| **Date:** | Tue, 10 Jun 2025 | **Prob (F-statistic):** | 1.44e-29 |
| **Time:** | 20:58:21 | **Log-Likelihood:** | 140.35 |
| **No. Observations:** | 94 | **AIC:** | -276.7 |
| **Df Residuals:** | 92 | **BIC:** | -271.6 |
| **Df Model:** | 1 | | |
| **Covariance Type:** | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **Intercept** | 0.0080 | 0.028 | 0.288 | 0.774 | -0.047 | 0.063 |
| **pyth_away** | 0.9239 | 0.055 | 16.683 | 0.000 | 0.814 | 1.034 |

| | | | |
|---|---|---|---|
| **Omnibus:** | 0.580 | **Durbin-Watson:** | 1.772 |
| **Prob(Omnibus):** | 0.748 | **Jarque-Bera (JB):** | 0.292 |
| **Skew:** | -0.126 | **Prob(JB):** | 0.864 |
| **Kurtosis:** | 3.105 | **Cond. No.** | 12.1 |

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- Se busca r-squared cercana a 1 para indicar que el modelo es bueno
- Se busca std err << coef
- Se busca t > 2 para indicar que el coeficiente es significativo
- Se busca P|t| < 0.05 para indicar que el coeficiente es significativo

## Análisis de una temporada en especifico

En esta seccion se calcula el pythagorean expectation por cada juego y se compara con el porcentaje de victorias real del equipo

In [13]:
```
def obtenerEstadisticasEquipo(team, temporada):
    query = text("""select extract(year from g.primer_lanzamiento)::integer as temporada, '{0}' as team,
                loc.nombre as home_team, g.carreras_local as home_r, g.carreras_visitante as vis_r,
                tj.descripcion as tipo_juego
                from juego g
                join equipo loc on g.local_id = loc.equipo_id
                join equipo vis on g.visitante_id = vis.equipo_id
                join tipo_juego tj on g.tipo_juego_id = tj.tipo_juego_id
                where (loc.nombre = '{0}' or
                vis.nombre = '{0}') and
                temporada = '{1}'""".format(team, temporada))
    with engine.connect() as conn:
        equipo = pd.read_sql(query, conn)

    # se agregan estas columnas para poder aplicar funciones de agregación
    equipo['hwin'] = np.where((equipo['home_r'] > equipo['vis_r']) & (equipo['home_team'] == team), 1, 0)
    equipo['hwin'] = equipo['hwin'].cumsum()
    equipo['home_r_h'] = np.where(equipo['home_team'] == team, equipo['home_r'], 0)
    equipo['vis_r_h'] = np.where(equipo['home_team'] == team, equipo['vis_r'], 0)
    equipo['G_h'] = np.where(equipo['home_team'] == team, 1, 0)
    equipo['G_h'] = equipo['G_h'].cumsum()
    equipo['awin'] = np.where((equipo['home_r'] < equipo['vis_r']) & (equipo['home_team'] != team), 1, 0)
    equipo['awin'] = equipo['awin'].cumsum()
    equipo['home_r_a'] = np.where(equipo['home_team'] != team, equipo['home_r'], 0)
    equipo['vis_r_a'] = np.where(equipo['home_team'] != team, equipo['vis_r'], 0)
    equipo['G_a'] = np.where(equipo['home_team'] != team, 1, 0)
    equipo['G_a'] = equipo['G_a'].cumsum()
    equipo['W_season'] = equipo['hwin'] + equipo['awin']
    equipo['G_season'] = np.arange(1, len(equipo) + 1)
    equipo['R_season'] = equipo['home_r_h'] + equipo['vis_r_a']
    equipo['R_season'] = equipo['R_season'].cumsum()
    equipo['R_home'] = equipo['home_r_h'].cumsum()
    equipo['R_away'] = equipo['vis_r_a'].cumsum()
    equipo['RA_season'] = equipo['vis_r_h'] + equipo['home_r_a']
    equipo['RA_season'] = equipo['RA_season'].cumsum()
    equipo['RA_home'] = equipo['vis_r_h'].cumsum()
    equipo['RA_away'] = equipo['home_r_a'].cumsum()
    equipo['wpct_season'] = round(equipo['W_season'] / equipo['G_season'], 3)
    equipo['wpct_home'] = round(equipo['hwin'] / equipo['G_h'], 3)
    equipo['wpct_away'] = round(equipo['awin'] / equipo['G_a'], 3)
    equipo['pyth_season'] = round((equipo['R_season'] ** 2) / (equipo['R_season'] ** 2 + equipo['RA_season']
** 2), 3)
```

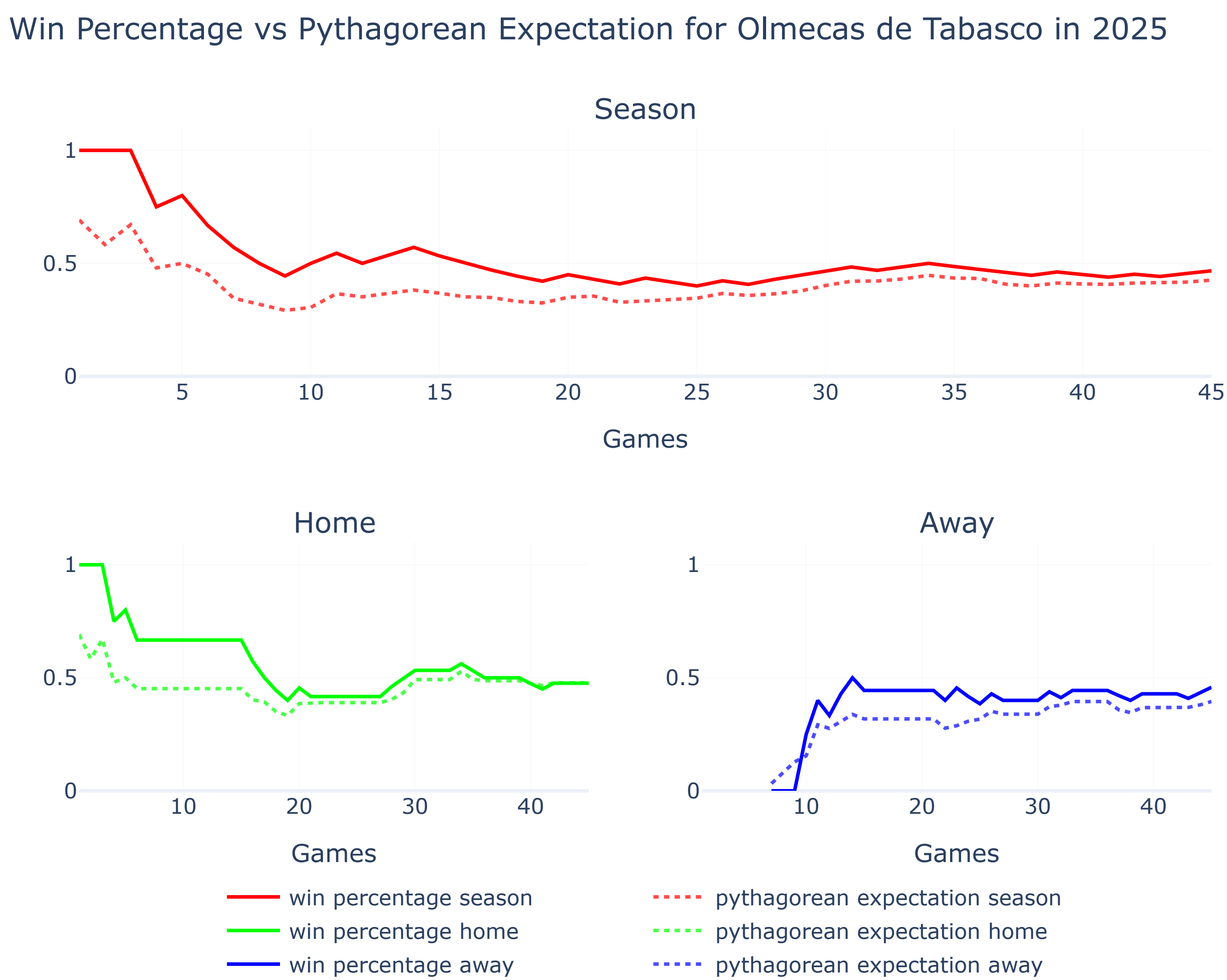
```
    equipo['pyth_home'] = round((equipo['R_home'] ** 2) / (equipo['R_home'] ** 2 + equipo['RA_home'] ** 2),
3)
    equipo['pyth_away'] = round((equipo['R_away'] ** 2) / (equipo['R_away'] ** 2 + equipo['RA_away'] ** 2),
3)
    equipo.drop(['home_team', 'home_r', 'vis_r'], axis=1, inplace=True)

    return equipo
```

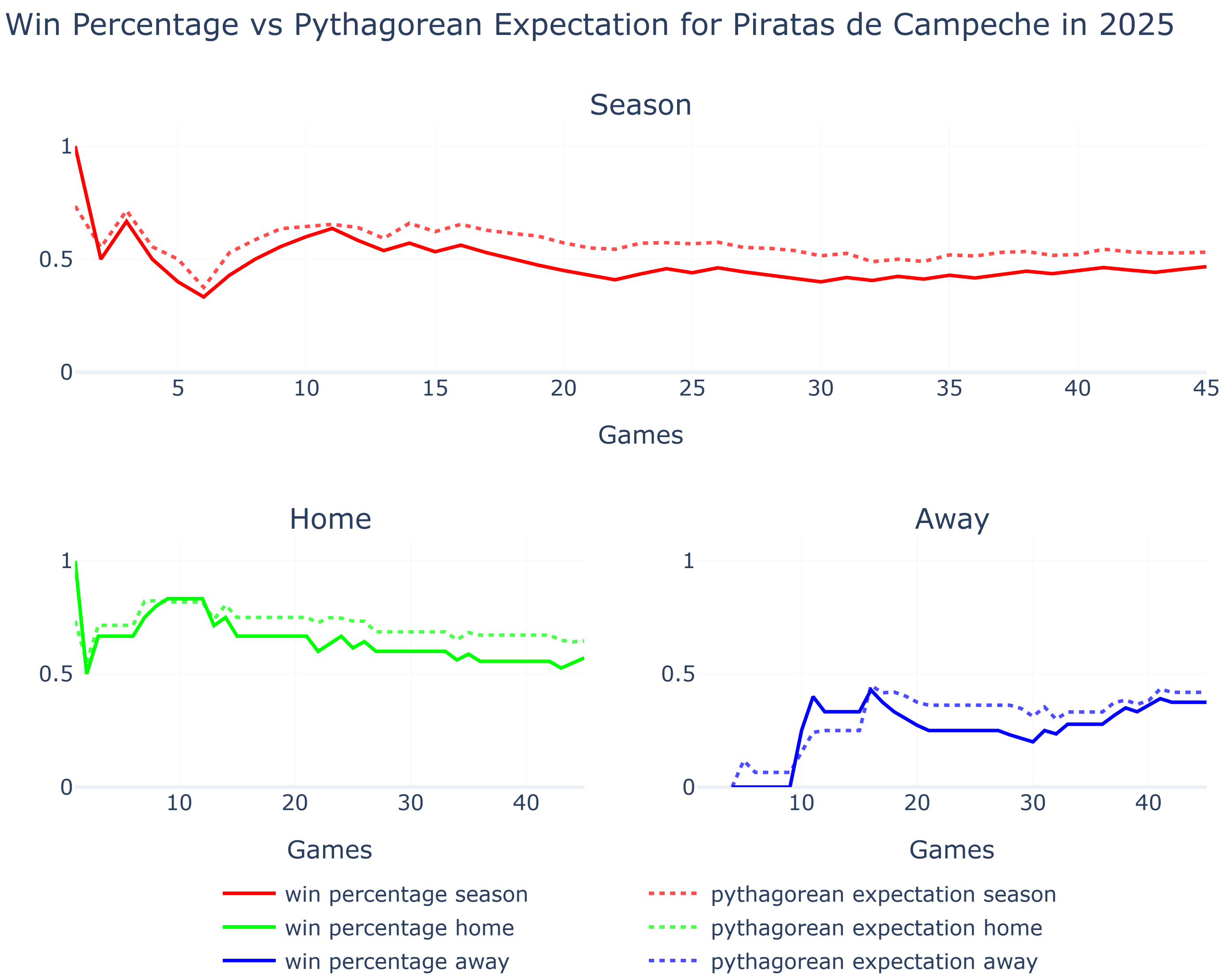
In [14]:
```
def graficarEstadisticasEquipo(equipo, team, temporada):
    fig = make_subplots(
        rows=2, cols=2,
        specs=[[{"colspan": 2}, None],
               [{}, {}]],
        subplot_titles=("Season","Home","Away"))

    fig.add_trace(go.Scatter(
        x=equipo['G_season'],
        y=equipo['wpct_season'],
        line=dict(color='rgba(255, 0, 0, 1)'),
        hovertext=equipo[['W_season', 'G_season', 'wpct_season', 'pyth_season']]
                        .apply(lambda x: f"""Wins: {x['W_season']}<br>Games:
                        {x['G_season']}<br>wpct: {x['wpct_season']}<br>Pyth: {x['pyth_season']}""", axis=
1),
        hoverinfo='text',
        name='win percentage season'
        ), row=1, col=1)

    fig.add_trace(go.Scatter(
        x=equipo['G_season'],
        y=equipo['pyth_season'],
        line=dict(color='rgba(255, 0, 0, 0.7)', dash='dot'),
        hovertext=equipo[['R_season', 'RA_season', 'wpct_season', 'pyth_season']]
                        .apply(lambda x: f"""Runs: {x['R_season']}<br>Allowed Runs:
                        {x['RA_season']}<br>wpct: {x['wpct_season']}<br>Pyth: {x['pyth_season']}""", axis
=1),
        hoverinfo='text',
        name='pythagorean expectation season'
        ), row=1, col=1)

    fig.add_trace(go.Scatter(
        x=equipo['G_season'],
        y=equipo['wpct_home'],
        line=dict(color='rgba(0, 255, 0, 1)'),
        hovertext=equipo[['hwin', 'G_h', 'wpct_home', 'pyth_home']]
                        .apply(lambda x: f"""Wins: {x['hwin']}<br>Games:
                        {x['G_h']}<br>wpct: {x['wpct_home']}<br>Pyth: {x['pyth_home']}""", axis=1),
        hoverinfo='text',
        name='win percentage home'
        ), row=2, col=1)
```

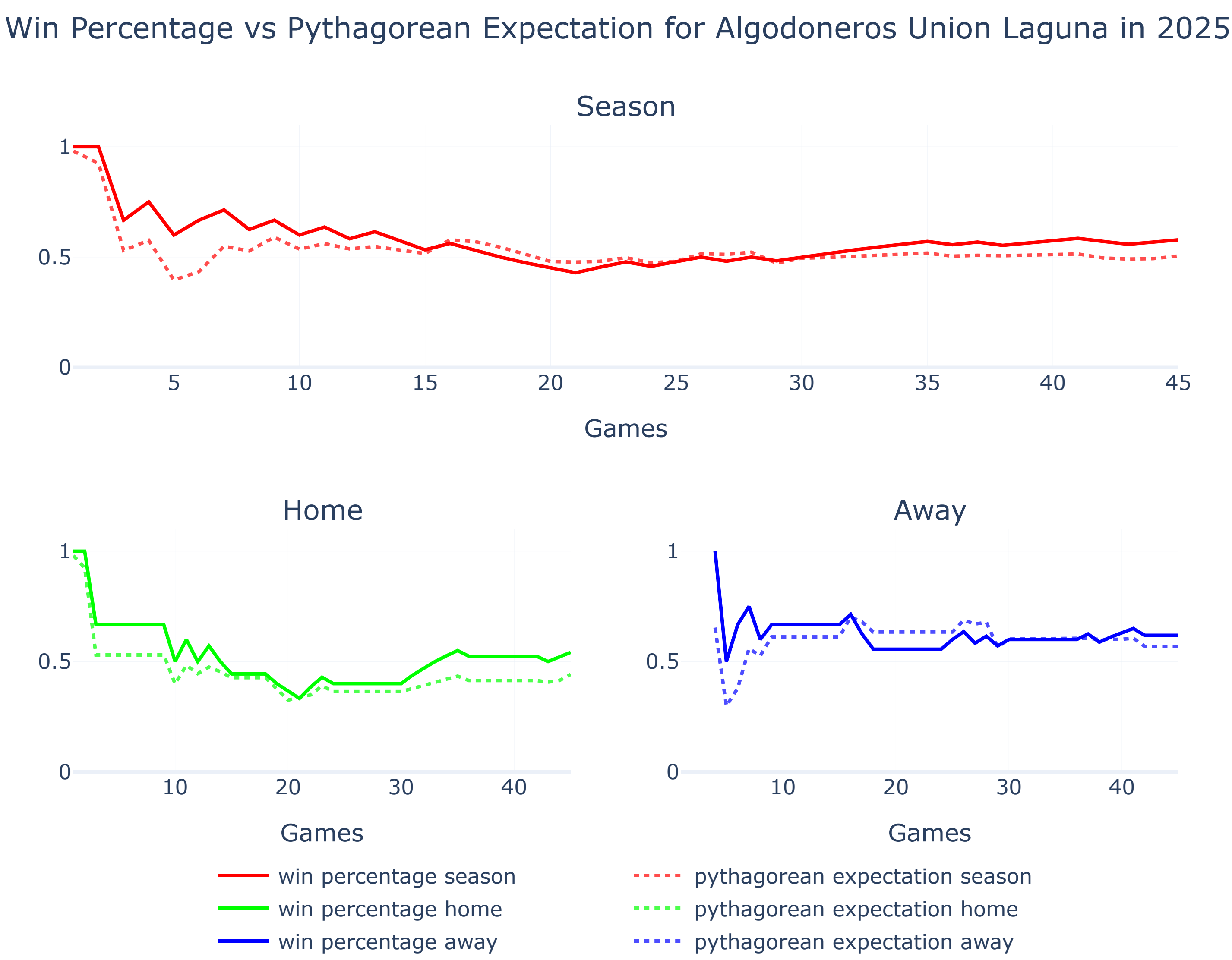
```
    fig.add_trace(go.Scatter(
        x=equipo['G_season'],
        y=equipo['pyth_home'],
        line=dict(color='rgba(0, 255, 0, 0.7)', dash='dot'),
        hovertext=equipo[['R_home', 'RA_home', 'wpct_home', 'pyth_home']]
                    .apply(lambda x: f"""Runs: {x['R_home']}<br>Allowed Runs:
                    {x['RA_home']}<br>wpct: {x['wpct_home']}<br>Pyth: {x['pyth_home']}""", axis=1),
        hoverinfo='text',
        name='pythagorean expectation home'
        ), row=2, col=1)

    fig.add_trace(go.Scatter(
        x=equipo['G_season'],
        y=equipo['wpct_away'],
        line=dict(color='rgba(0, 0, 255, 1)'),
        hovertext=equipo[['awin', 'G_a', 'wpct_away', 'pyth_away']]
                    .apply(lambda x: f"""Wins: {x['awin']}<br>Games:
                    {x['G_a']}<br>wpct: {x['wpct_away']}<br>Pyth: {x['pyth_away']}""", axis=1),
        hoverinfo='text',
        name='win percentage away'
        ), row=2, col=2)

    fig.add_trace(go.Scatter(
        x=equipo['G_season'],
        y=equipo['pyth_away'],
        line=dict(color='rgba(0, 0, 255, 0.7)', dash='dot'),
        hovertext=equipo[['R_away', 'RA_away', 'wpct_away', 'pyth_away']]
                    .apply(lambda x: f"""Runs: {x['R_away']}<br>Allowed Runs:
                    {x['RA_away']}<br>wpct: {x['wpct_away']}<br>Pyth: {x['pyth_away']}""", axis=1),
        hoverinfo='text',
        name='pythagorean expectation away'
        ), row=2, col=2)

    fig.update_layout(
        title=f'Win Percentage vs Pythagorean Expectation for {team} in {temporada}',
        width=800,
        height=600,
        showlegend=True,
        template='plotly_white',
        legend=dict(
            orientation="h",
            yanchor="bottom",
            y=-0.3,
```

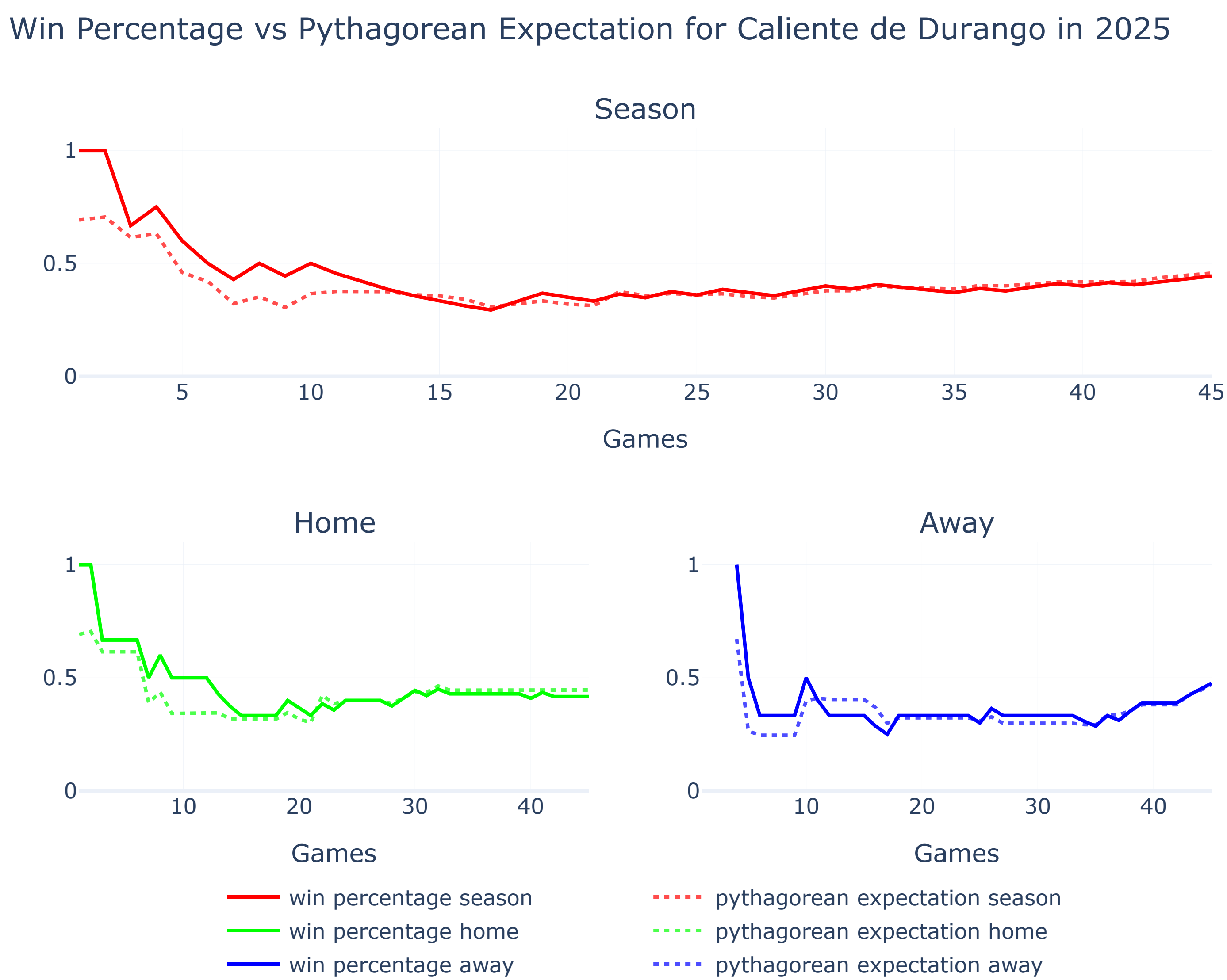
```
            xanchor="center",
            x=0.5
        ),
    )

    fig.update_xaxes(title_text='Games', row=1, col=1)
    fig.update_xaxes(title_text='Games', row=2, col=1)
    fig.update_xaxes(title_text='Games', row=2, col=2)
    fig.update_yaxes(range=[0, 1.1], row=1, col=1)
    fig.update_yaxes(range=[0, 1.1], row=2, col=1)
    fig.update_yaxes(range=[0, 1.1], row=2, col=2)
    return fig
```

In [15]:
```
temporada = 2025
query = text("""select distinct(e.nombre) as team
                from juego g
                join equipo e on g.local_id = e.equipo_id
                where g.temporada = '{0}'""".format(temporada))
equipos = pd.read_sql(query, engine)['team'].tolist()
for team in equipos:
    df = obtenerEstadisticasEquipo(team, temporada)
    graficarEstadisticasEquipo(df, team, temporada).show()
```

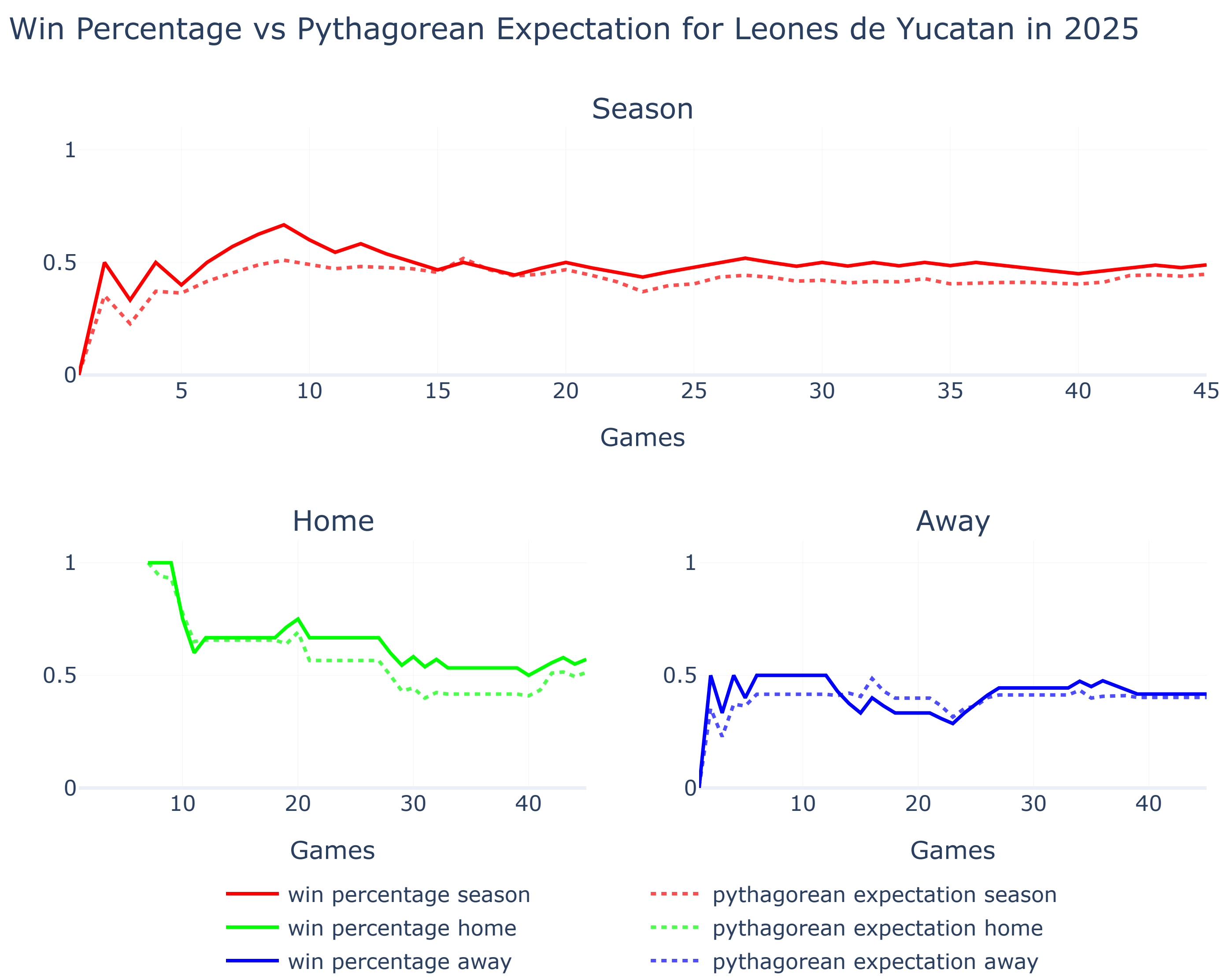

Win Percentage vs Pythagorean Expectation for Olmecas de Tabasco in 2025
Season
Home
Away
Games
Games
Games
1
0.5
0
5
10
15
20
25
30
35
40
45
10
20
30
40
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

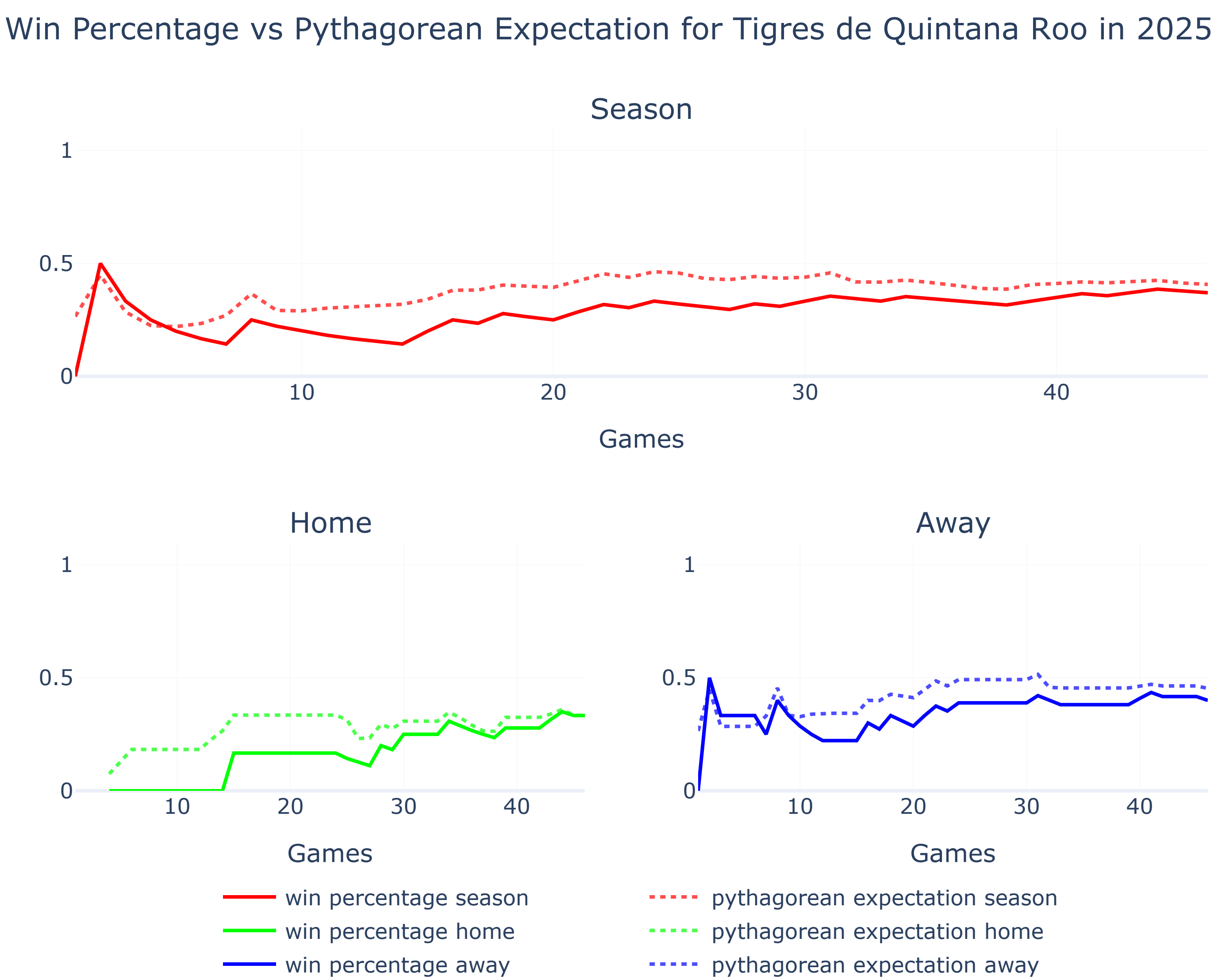

Win Percentage vs Pythagorean Expectation for Piratas de Campeche in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away


Win Percentage vs Pythagorean Expectation for Algodoneros Union Laguna in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

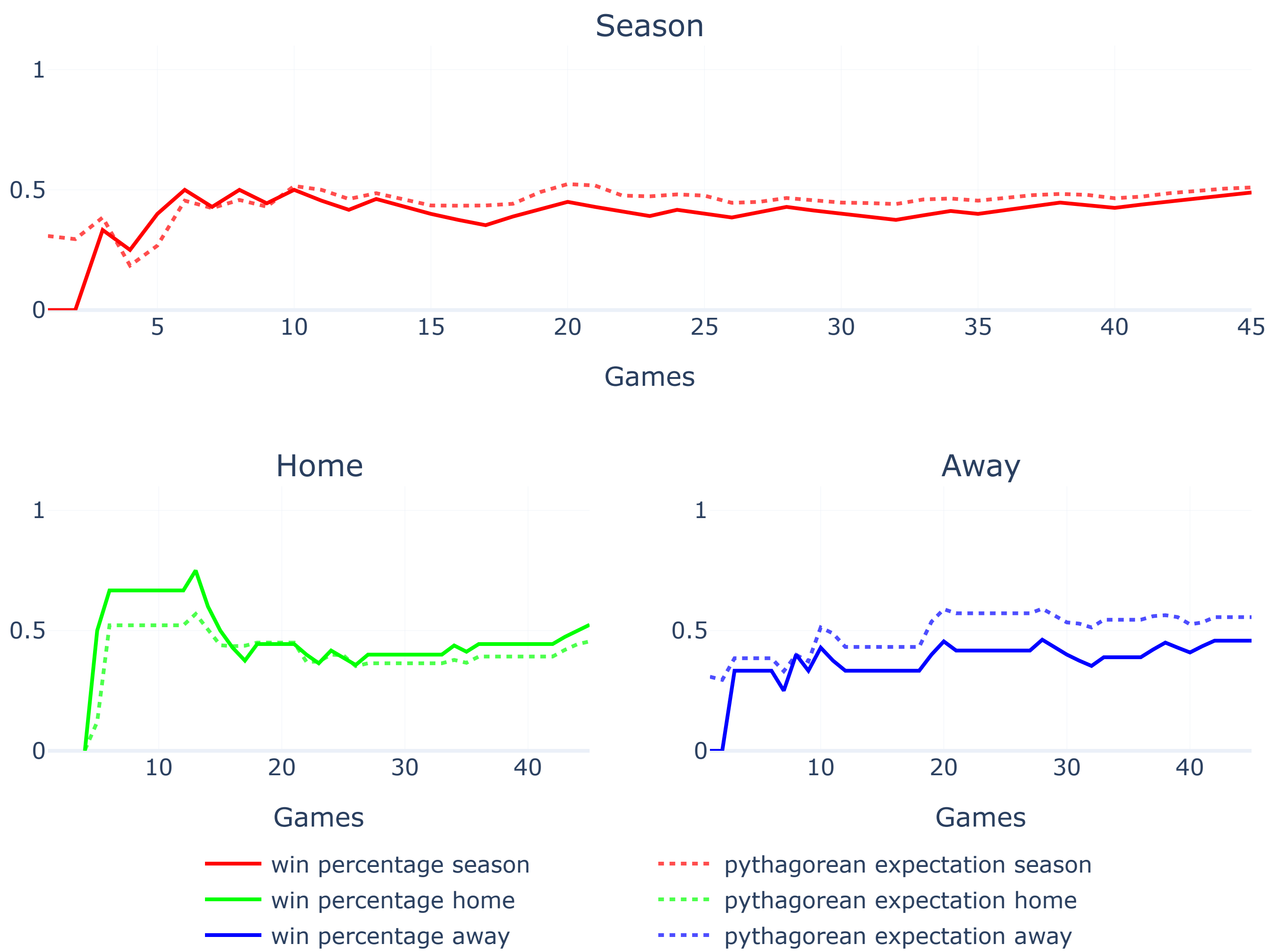


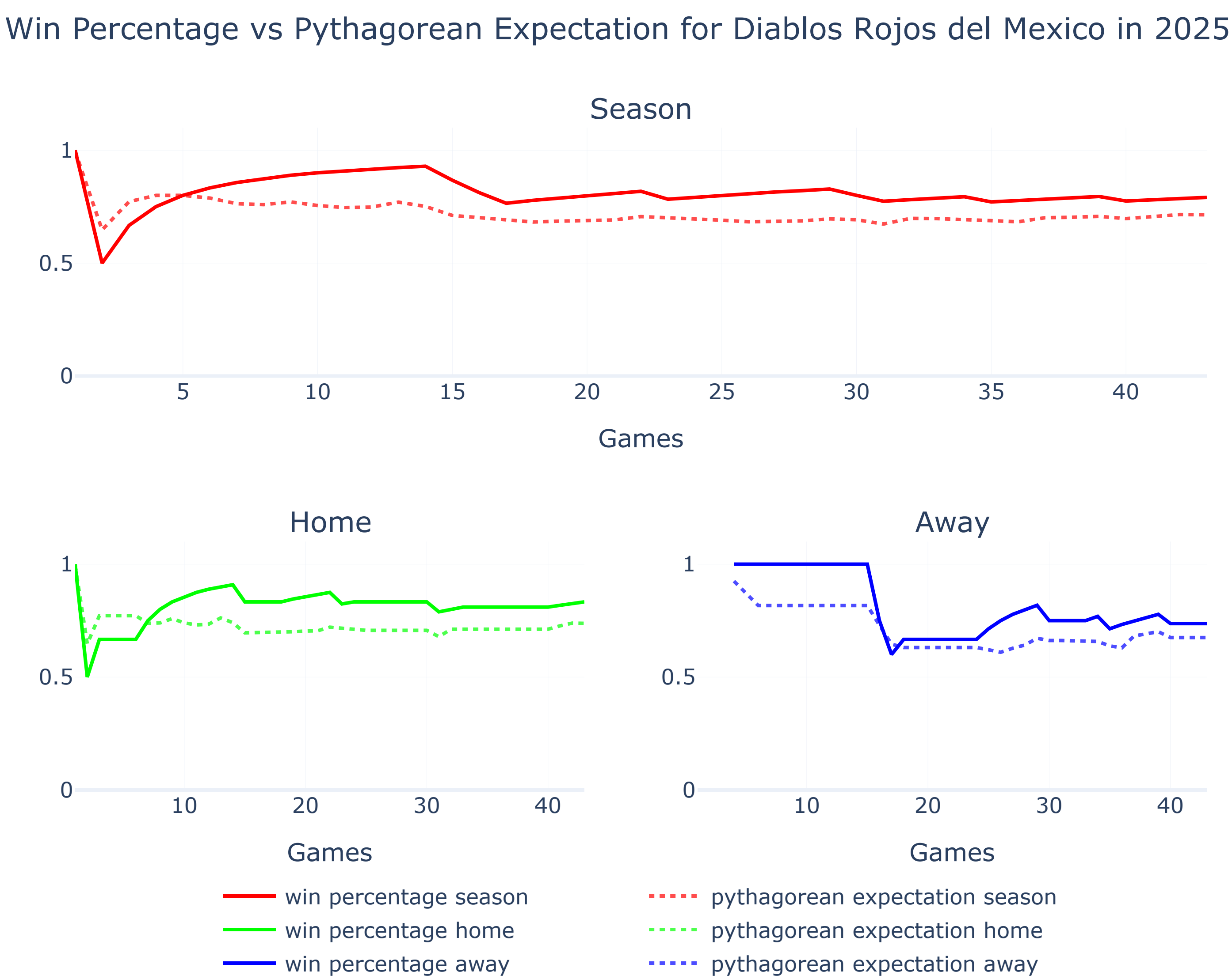
Win Percentage vs Pythagorean Expectation for Caliente de Durango in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

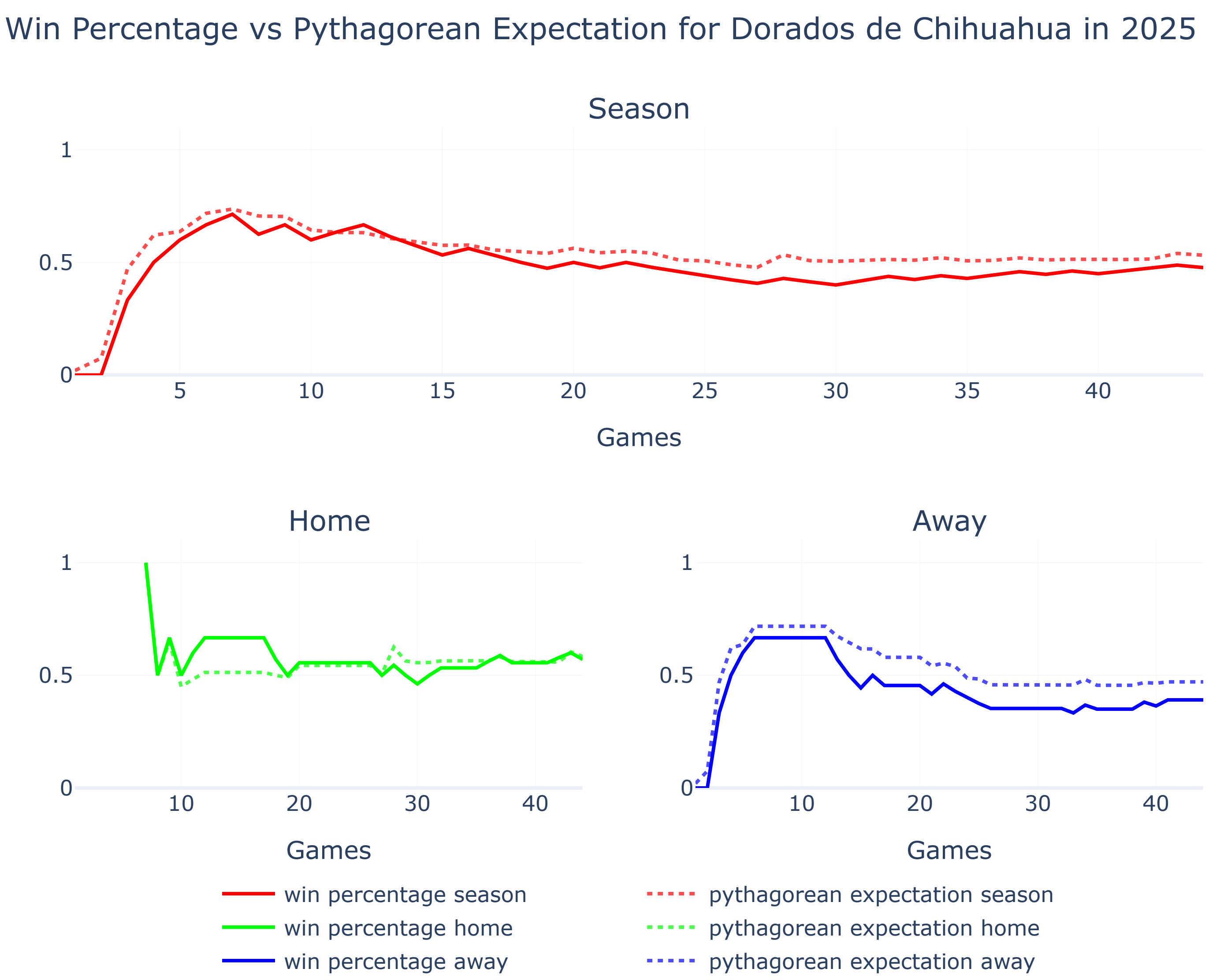

Win Percentage vs Pythagorean Expectation for Leones de Yucatan in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

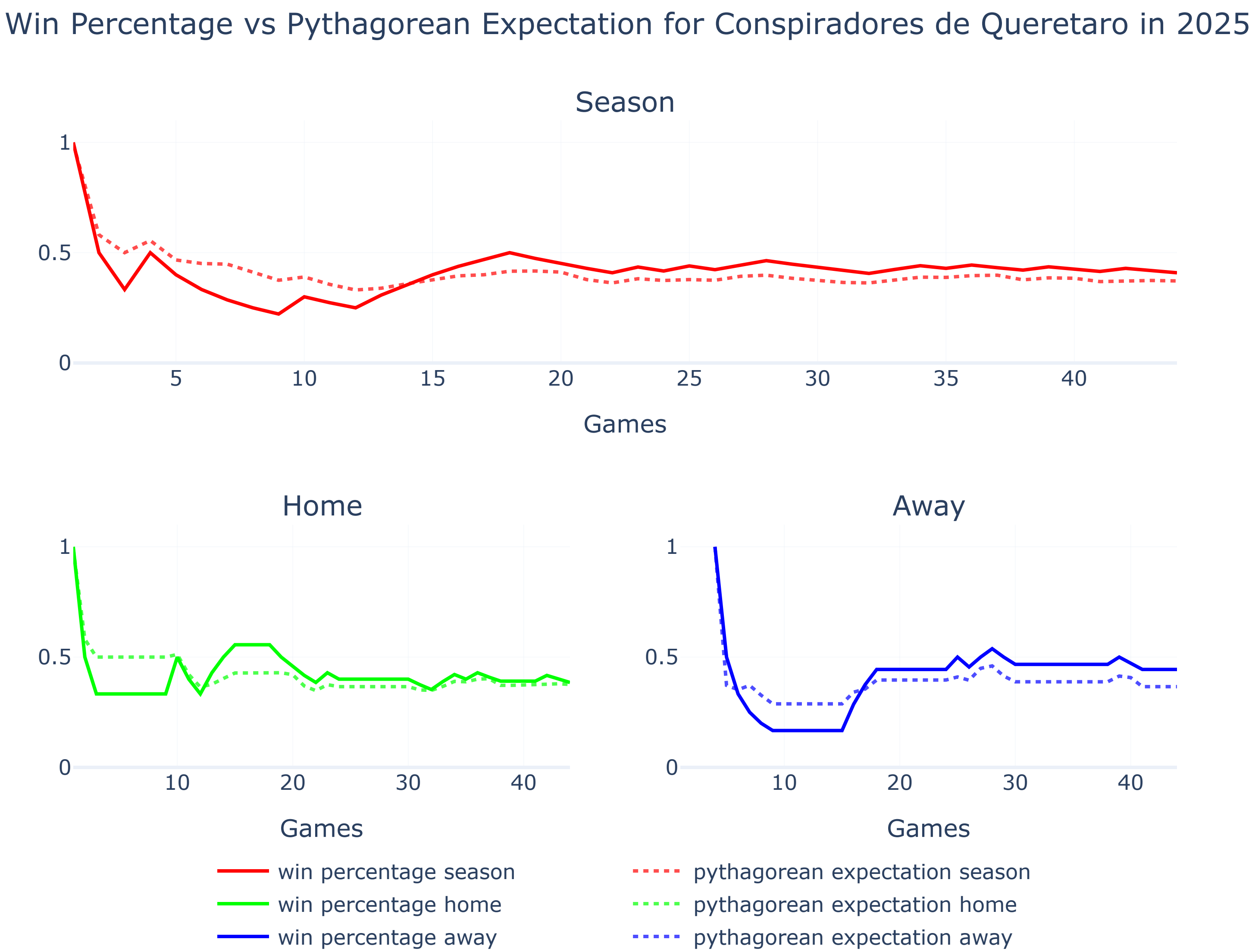

Win Percentage vs Pythagorean Expectation for Tigres de Quintana Roo in 2025
Season
1
0.5
0
10
20
30
40
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

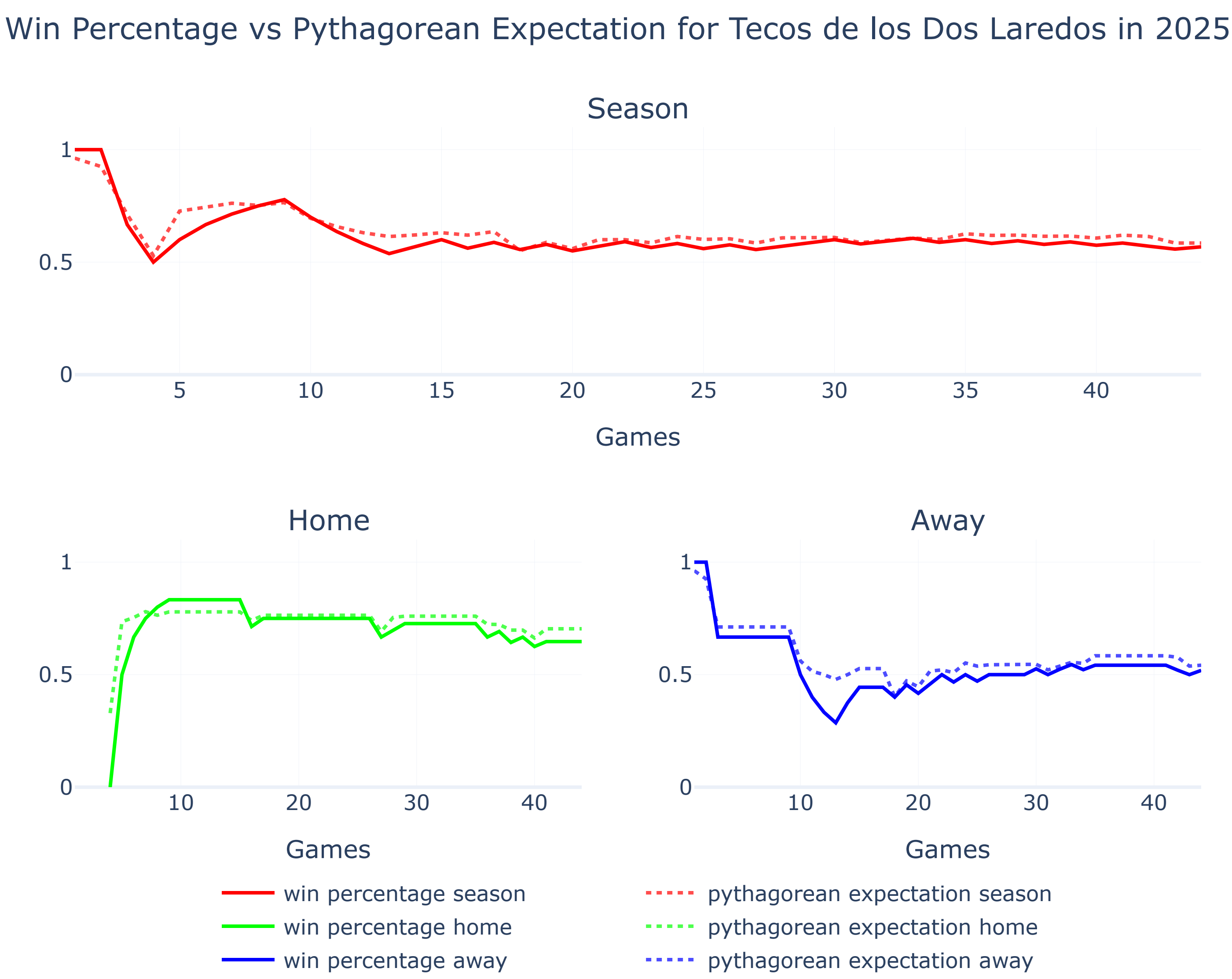
# Win Percentage vs Pythagorean Expectation for Saraperos de Saltillo in 2025

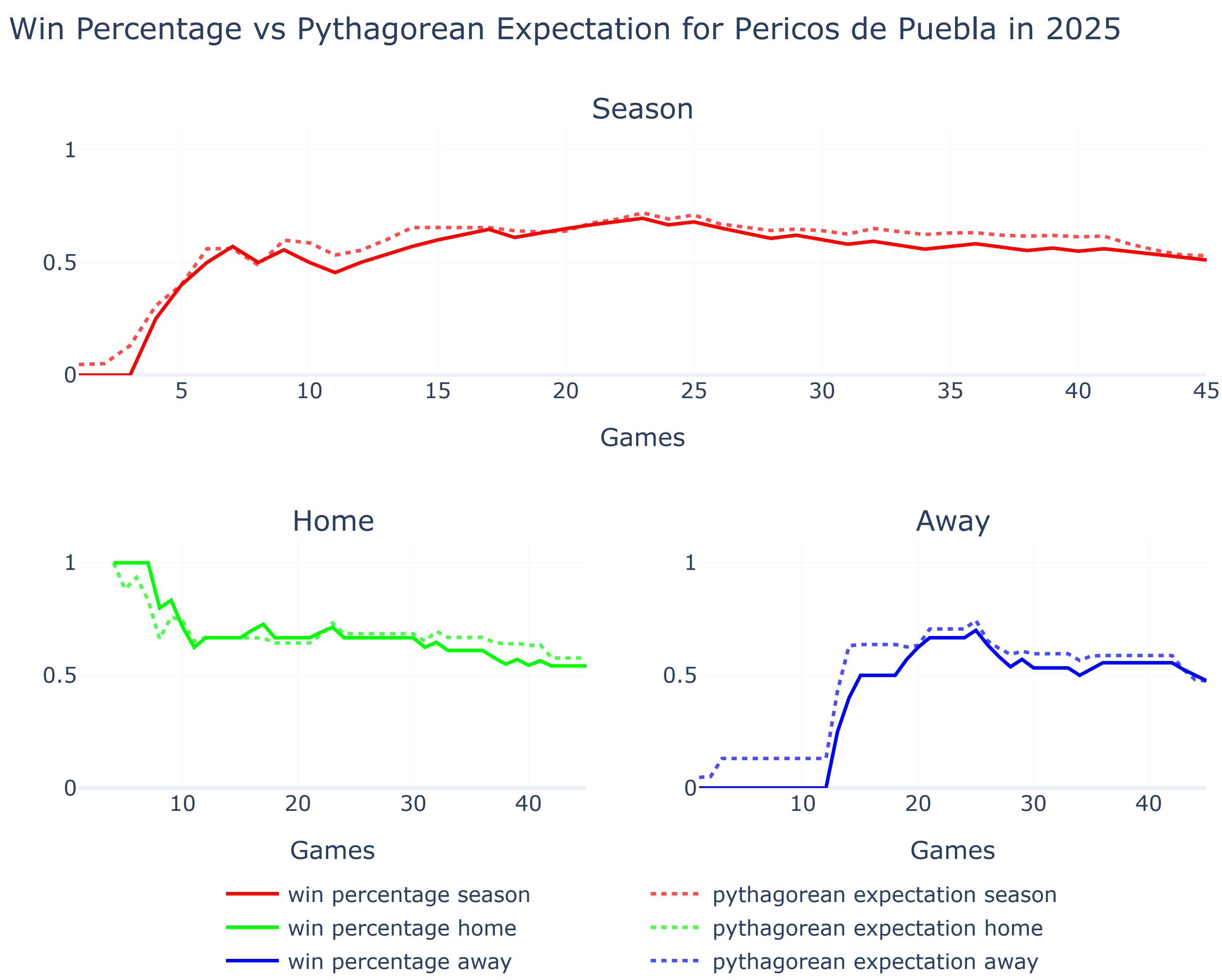


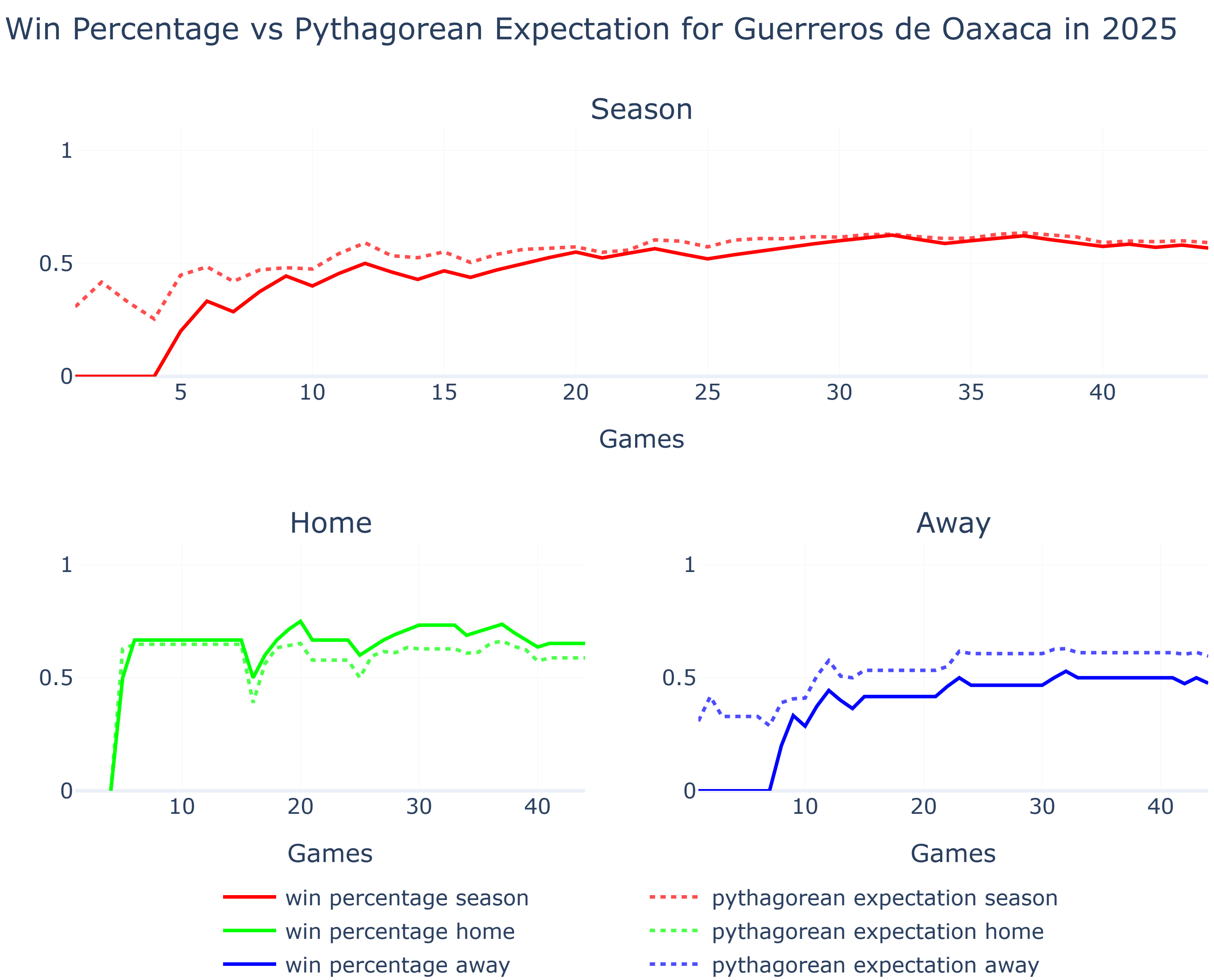

Win Percentage vs Pythagorean Expectation for Diablos Rojos del Mexico in 2025
Season
Home
Away
Games
Games
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

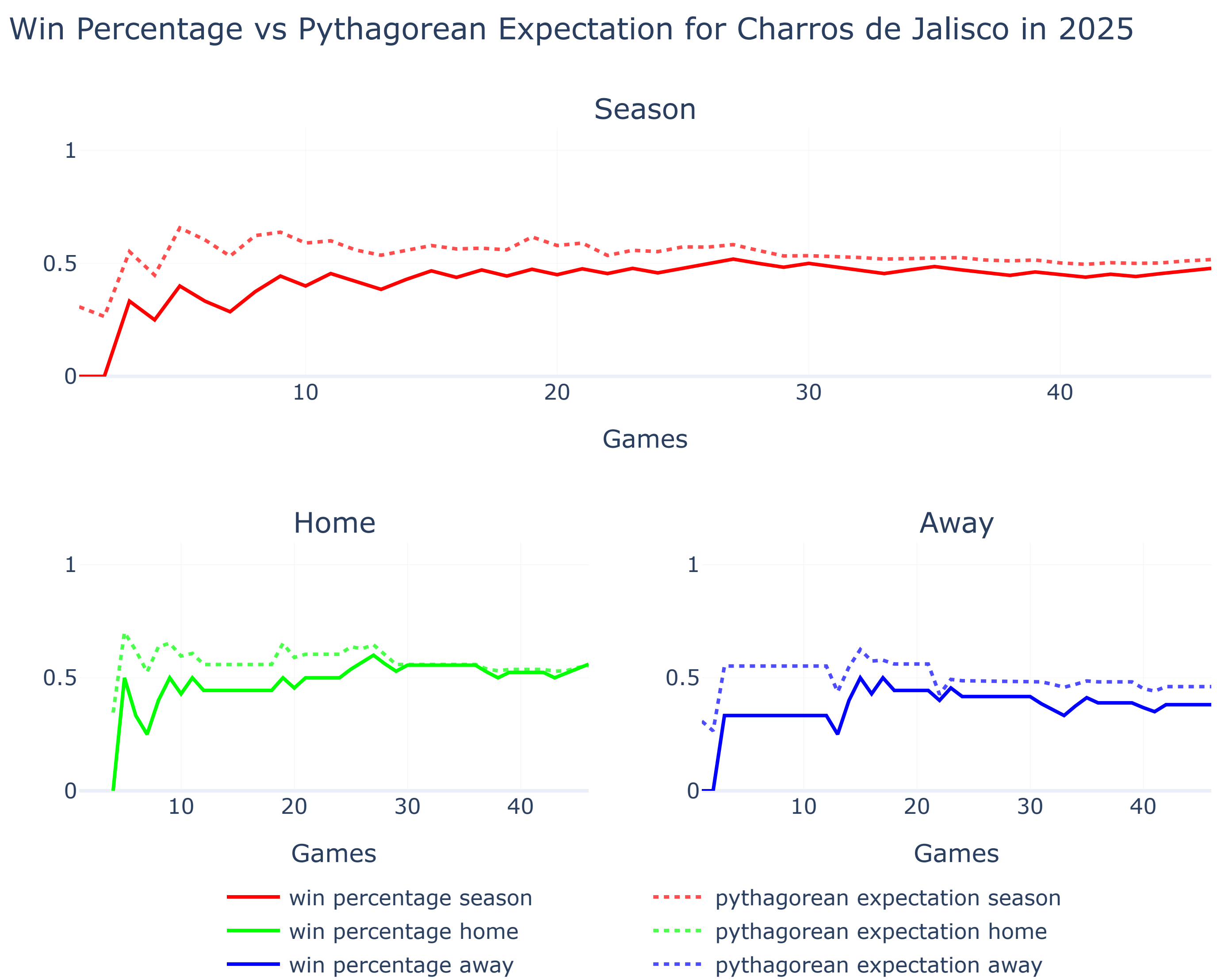

Win Percentage vs Pythagorean Expectation for Dorados de Chihuahua in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

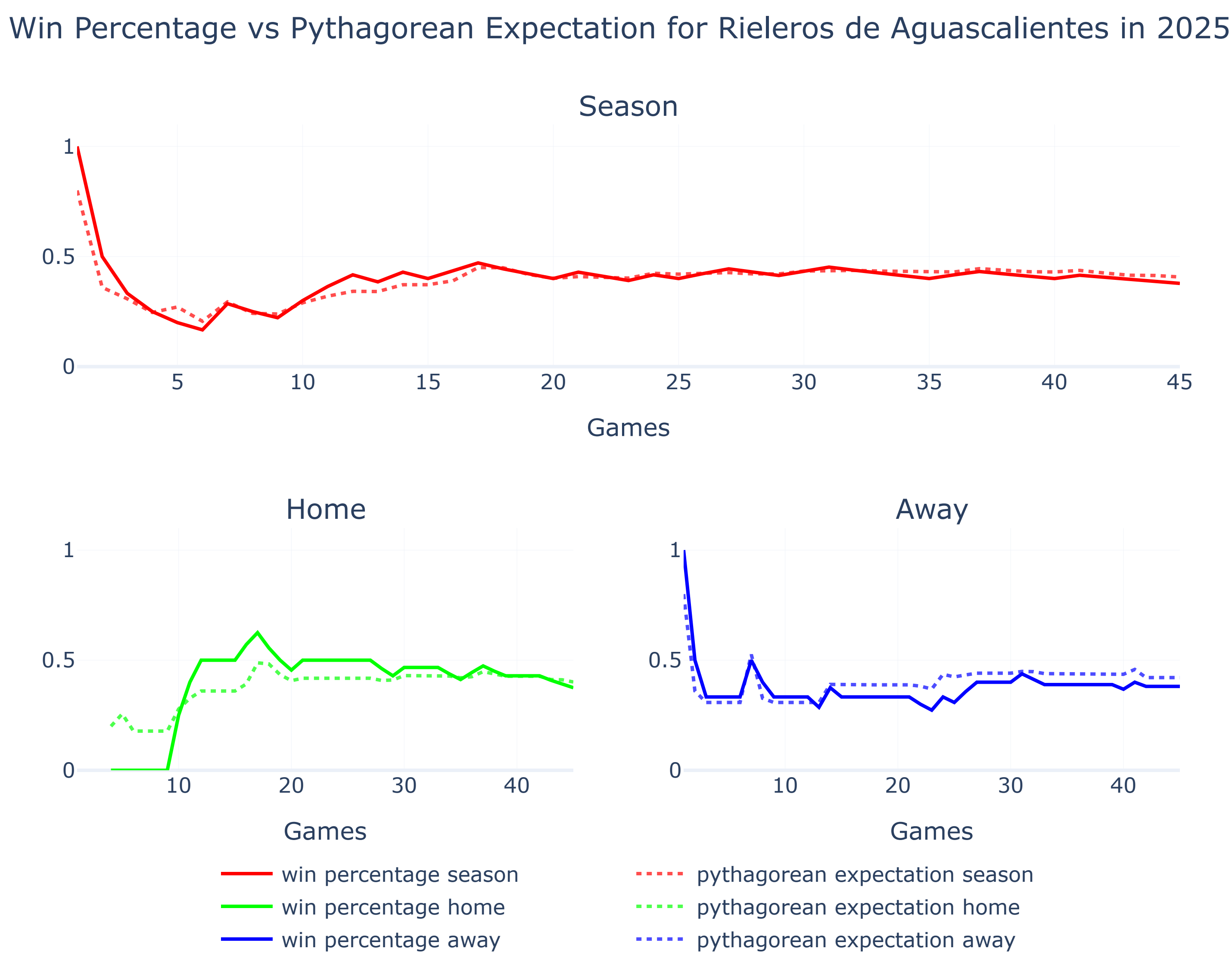

Win Percentage vs Pythagorean Expectation for Conspiradores de Queretaro in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

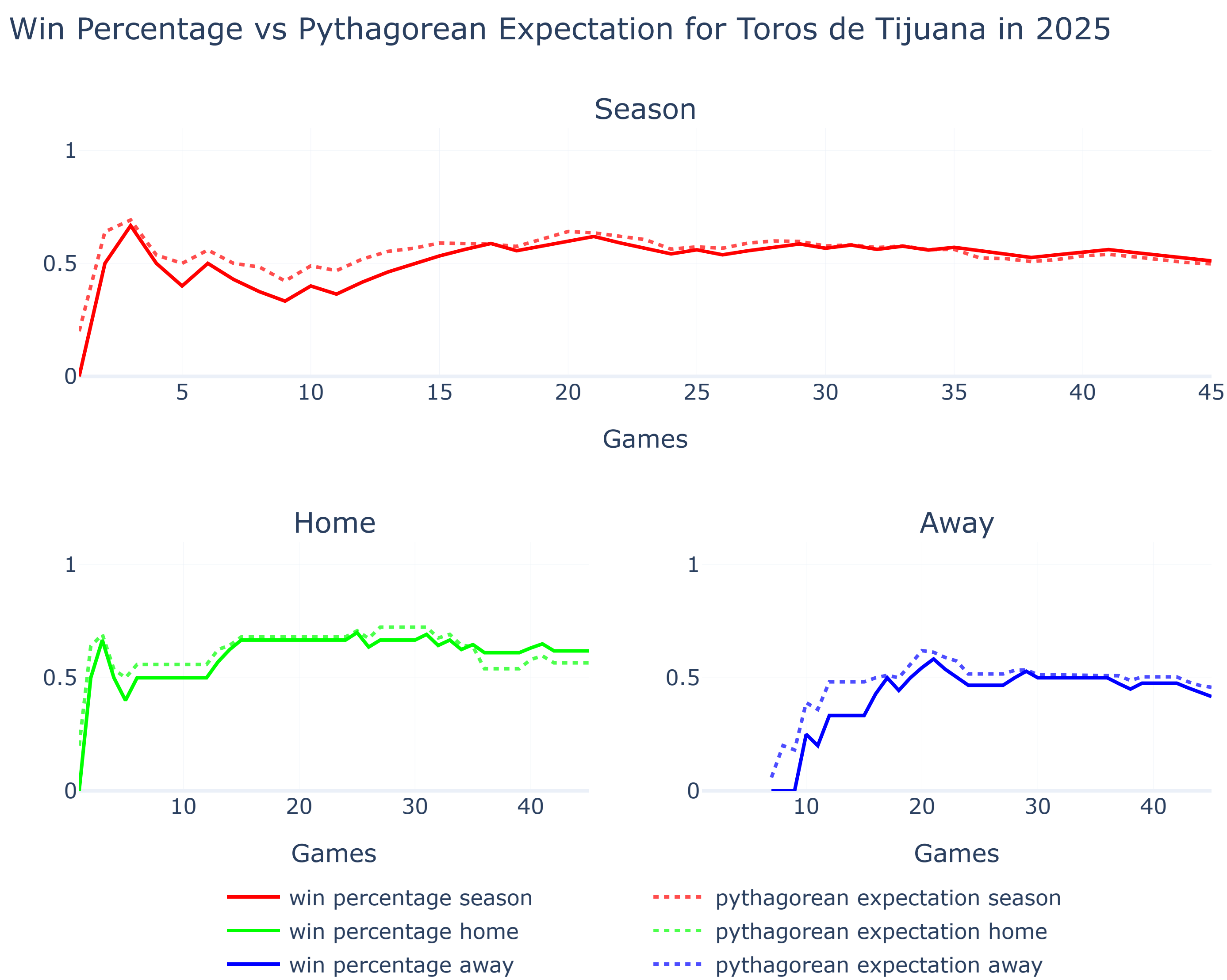

Win Percentage vs Pythagorean Expectation for Tecos de los Dos Laredos in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
pythagorean expectation season
win percentage home
pythagorean expectation home
win percentage away
pythagorean expectation away

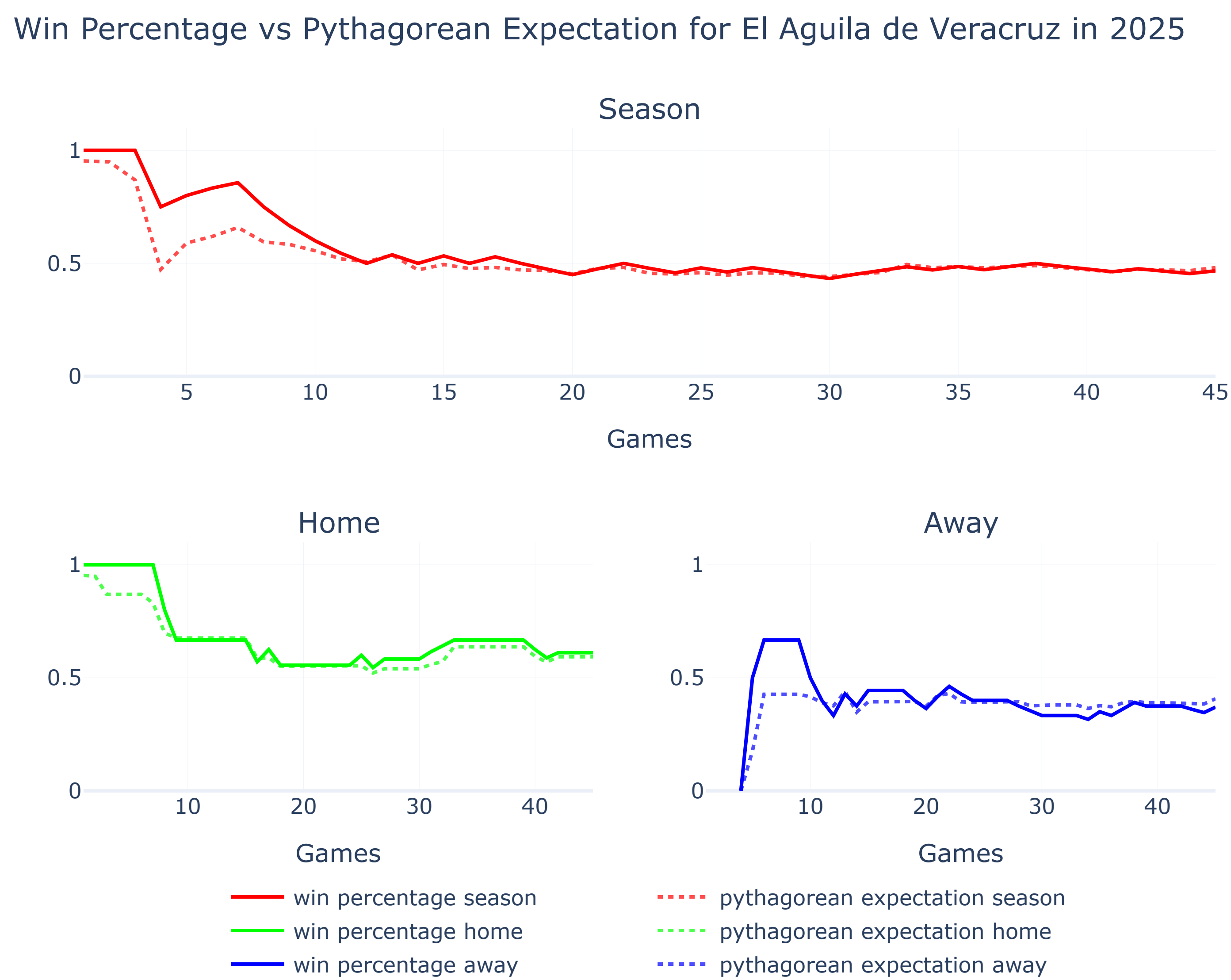

Win Percentage vs Pythagorean Expectation for Pericos de Puebla in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

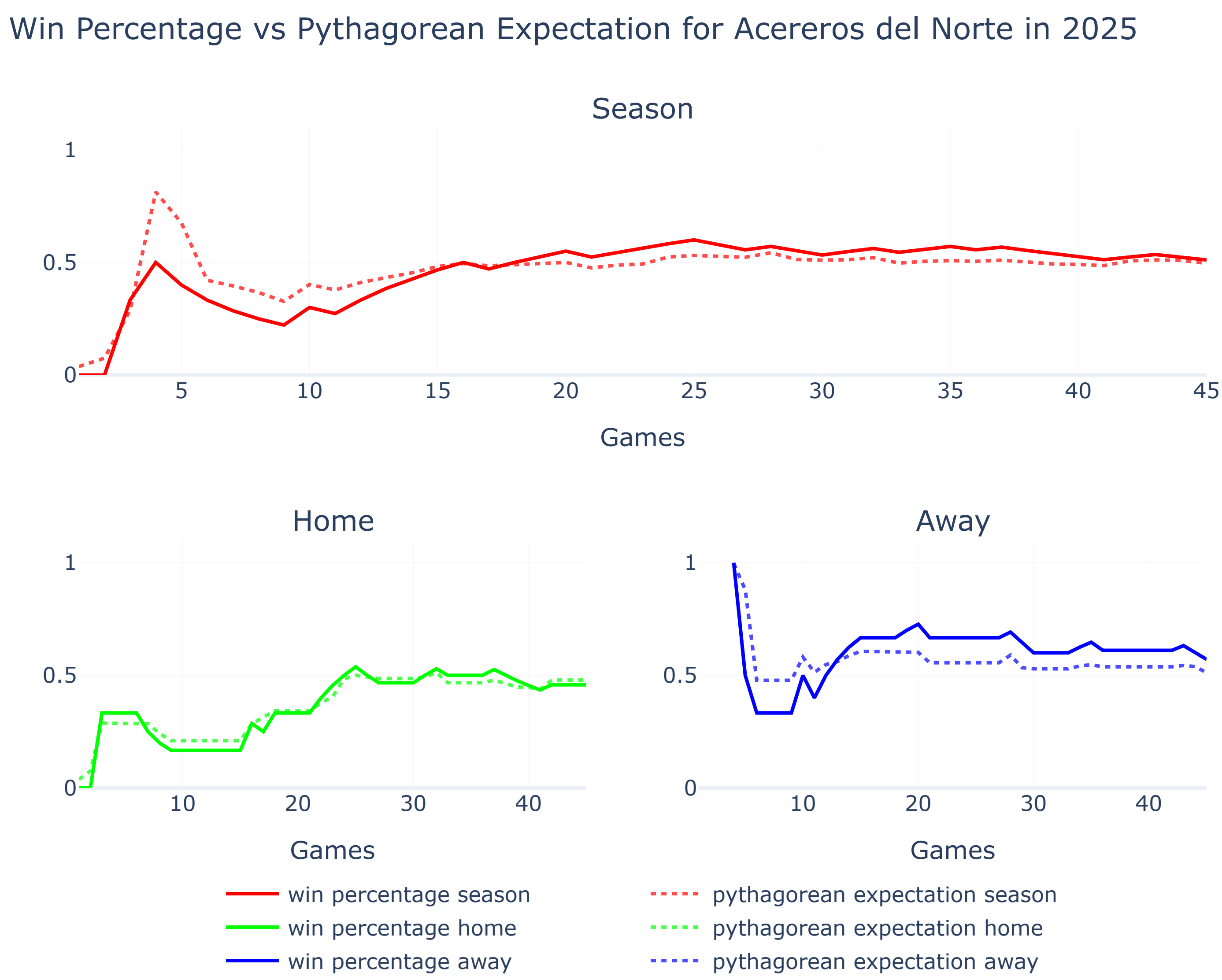

Win Percentage vs Pythagorean Expectation for Guerreros de Oaxaca in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

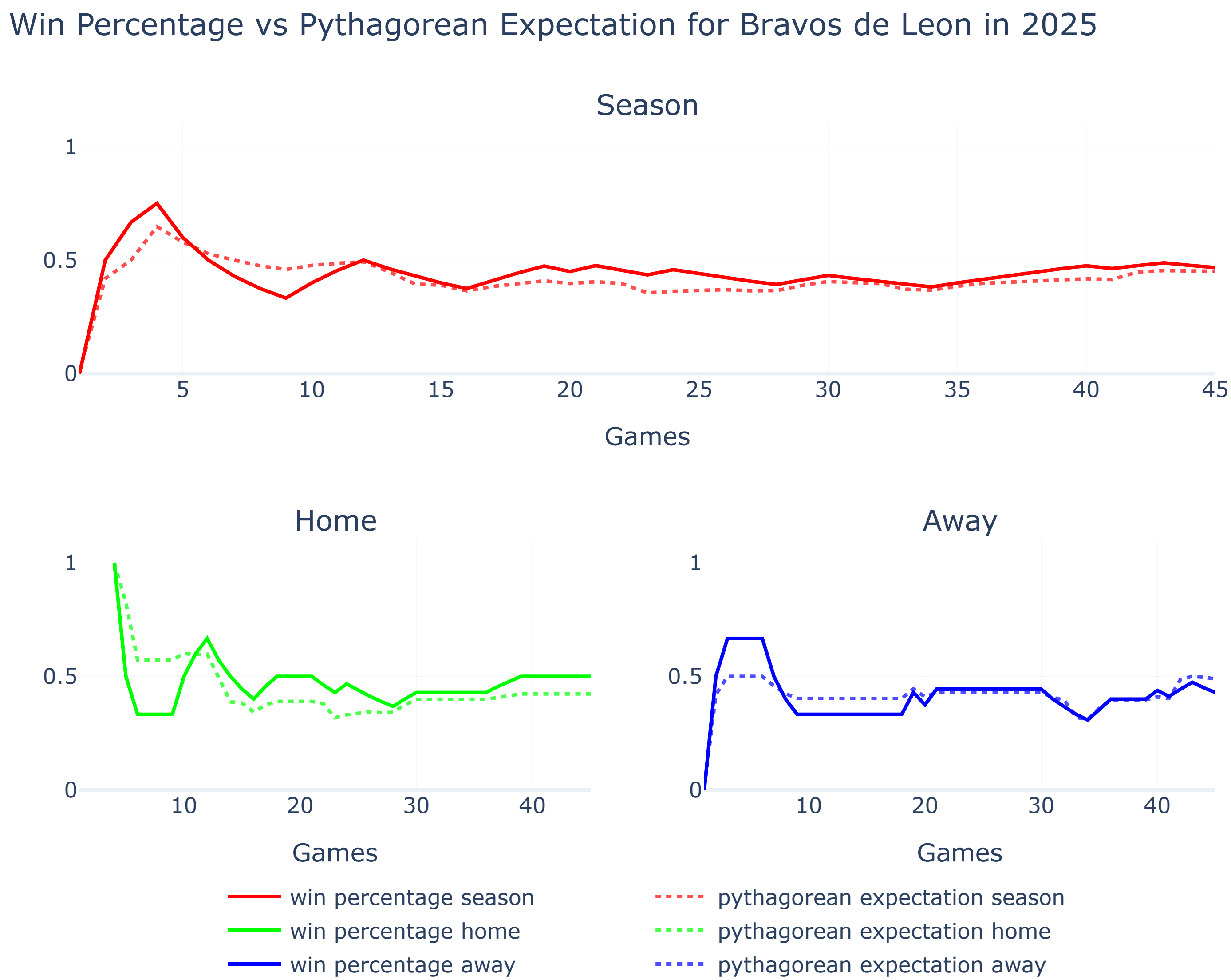

Win Percentage vs Pythagorean Expectation for Charros de Jalisco in 2025
Season
1
0.5
0
10
20
30
40
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

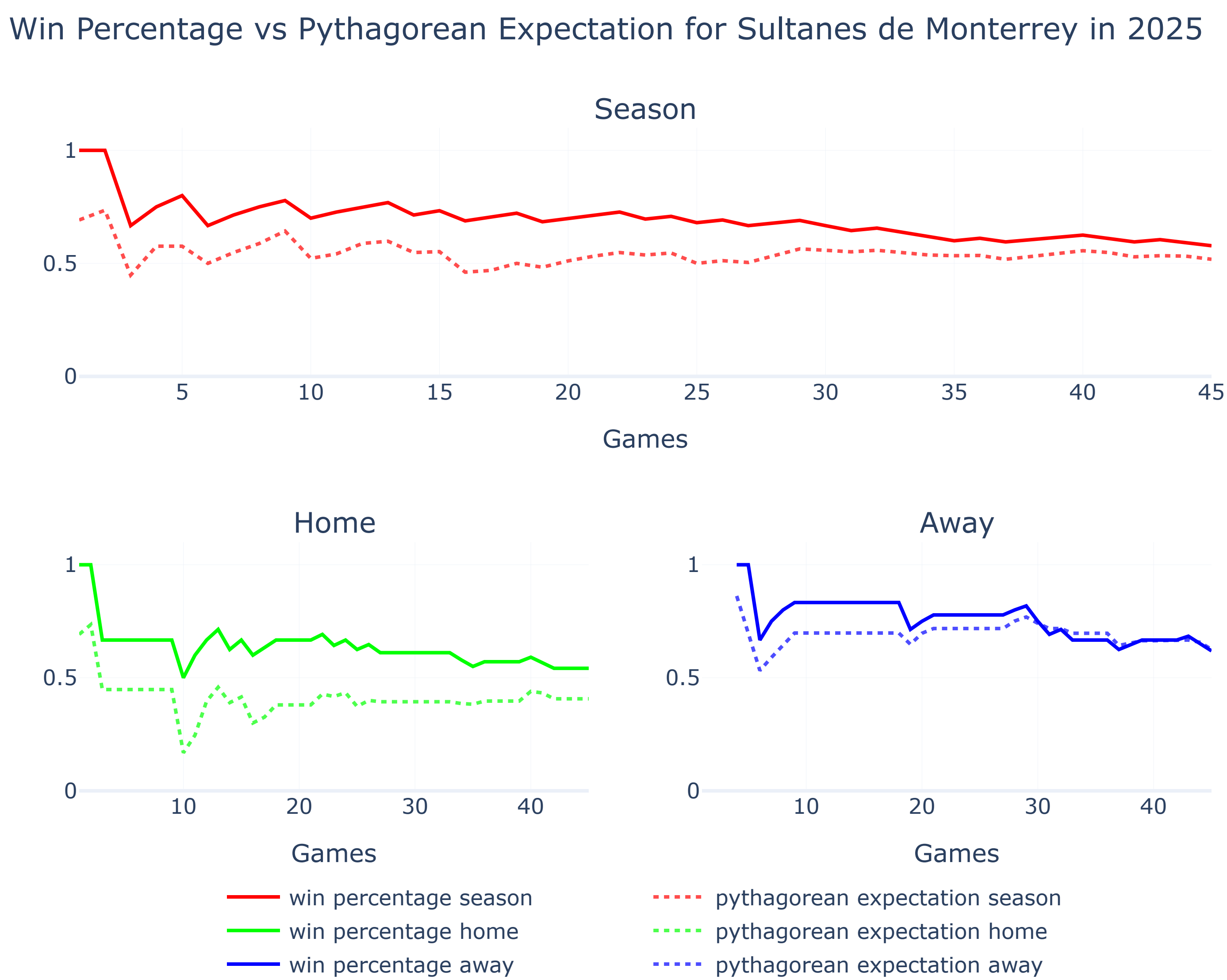

Win Percentage vs Pythagorean Expectation for Rieleros de Aguascalientes in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away


Win Percentage vs Pythagorean Expectation for Toros de Tijuana in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
Away
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away


Win Percentage vs Pythagorean Expectation for El Aguila de Veracruz in 2025
Season
Home
Away
Games
Games
Games
1
0.5
0
5
10
15
20
25
30
35
40
45
10
20
30
40
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away


Win Percentage vs Pythagorean Expectation for Acereros del Norte in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away


Win Percentage vs Pythagorean Expectation for Bravos de Leon in 2025
Season
Home
Away
Games
0
0.5
1
5
10
15
20
25
30
35
40
45
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away


Win Percentage vs Pythagorean Expectation for Sultanes de Monterrey in 2025
Season
1
0.5
0
5
10
15
20
25
30
35
40
45
Games
Home
1
0.5
0
10
20
30
40
Games
Away
1
0.5
0
10
20
30
40
Games
win percentage season
win percentage home
win percentage away
pythagorean expectation season
pythagorean expectation home
pythagorean expectation away

# Uso como herramienta de pronóstico

Ahora vamos a vailidar si es una buena herramienta para pronosticar el rendimiento futuro en una temporada. En este caso se selecciona la temporada 2024 y se divide en el juego de las estrellas (24-mayo-2024). En los datos solo se Toman los juegos de temporada regular para poder comparar con el rendimiento real del equipo.

In [16]:
```
query = text("""select date_trunc('day', g.primer_lanzamiento)::date as date, loc.nombre as home_team,
              vis.nombre as visiting_team, g.carreras_local as home_r, g.carreras_visitante as vis_r
              from juego g
              join equipo loc on g.local_id = loc.equipo_id
              join equipo vis on g.visitante_id = vis.equipo_id
              where g.temporada = '2024' and
              g.tipo_juego_id = 'R'""")
with engine.connect() as conn:
    lmb = pd.read_sql(query, conn)

# se agregan estas columnas para poder aplicar funciones de agregación
lmb['count'] = 1
```

Crear df separados de los partidos de local y visitante. Unirlos en un solo df

In [17]:
```
lmbTemporadaLocal = lmb[['home_team','home_r','vis_r','count','date']].copy()
lmbTemporadaLocal['home'] = 1
lmbTemporadaLocal = lmbTemporadaLocal.rename(columns={'home_team': 'team', 'home_r': 'R', 'vis_r': 'RA'})

lmbTemporadaAway = lmb[['visiting_team','home_r','vis_r','count','date']].copy()
lmbTemporadaAway['home'] = 0
lmbTemporadaAway = lmbTemporadaAway.rename(columns={'visiting_team': 'team', 'vis_r': 'R', 'home_r': 'RA'})
lmbTemporada = pd.concat([lmbTemporadaLocal, lmbTemporadaAway], ignore_index=True)
```

Agregar columna de victoria

In [18]:
```
lmbTemporada['win'] = np.where(lmbTemporada['R'] > lmbTemporada['RA'], 1, 0)
```

Separar en 2 dfs uno para la primera mitad de la temporada y otro para la segunda mitad

In [19]:
```
mitad = datetime.datetime(2024, 5, 24).date()
lmb1stHalf = lmbTemporada[lmbTemporada['date'] < mitad]
lmb2ndHalf = lmbTemporada[lmbTemporada['date'] >= mitad]
lmb1stHalf.describe(), lmb2ndHalf.describe()
```

Out[19]:
```
(                R          RA  count        home         win
 count  718.000000  718.000000  718.0  718.000000  718.000000
 mean     5.651811    5.651811    1.0    0.500000    0.500000
 std      4.104791    4.104791    0.0    0.500349    0.500349
 min      0.000000    0.000000    1.0    0.000000    0.000000
 25%      3.000000    3.000000    1.0    0.000000    0.000000
 50%      5.000000    5.000000    1.0    0.500000    0.500000
 75%      8.000000    8.000000    1.0    1.000000    1.000000
 max     24.000000   24.000000    1.0    1.000000    1.000000,
                  R           RA   count         home          win
 count  1092.000000  1092.000000  1092.0  1092.000000  1092.000000
 mean      5.352564     5.352564     1.0     0.500000     0.499084
 std       3.761845     3.761845     0.0     0.500229     0.500228
 min       0.000000     0.000000     1.0     0.000000     0.000000
 25%       3.000000     3.000000     1.0     0.000000     0.000000
 50%       5.000000     5.000000     1.0     0.500000     0.000000
 75%       7.000000     7.000000     1.0     1.000000     1.000000
 max      22.000000    22.000000     1.0     1.000000     1.000000)
```

Calcular las columnas de rendimiento para cada equipo en cada mitad de la temporada. Para identificar las columnas se les agrega en el nombre 1 o 2 dependiendo de la mitad de la temporada

In [20]:
```
performance1stHalf = lmb1stHalf.groupby('team')[['count', 'win', 'R', 'RA']].sum().reset_index()
performance1stHalf = performance1stHalf.rename(columns={'count': 'count1', 'win': 'win1', 'R': 'R1', 'RA': 'R
A1'})

performance2ndHalf = lmb2ndHalf.groupby('team')[['count', 'win', 'R', 'RA']].sum().reset_index()
performance2ndHalf = performance2ndHalf.rename(columns={'count': 'count2', 'win': 'win2', 'R': 'R2', 'RA': 'R
A2'})
```

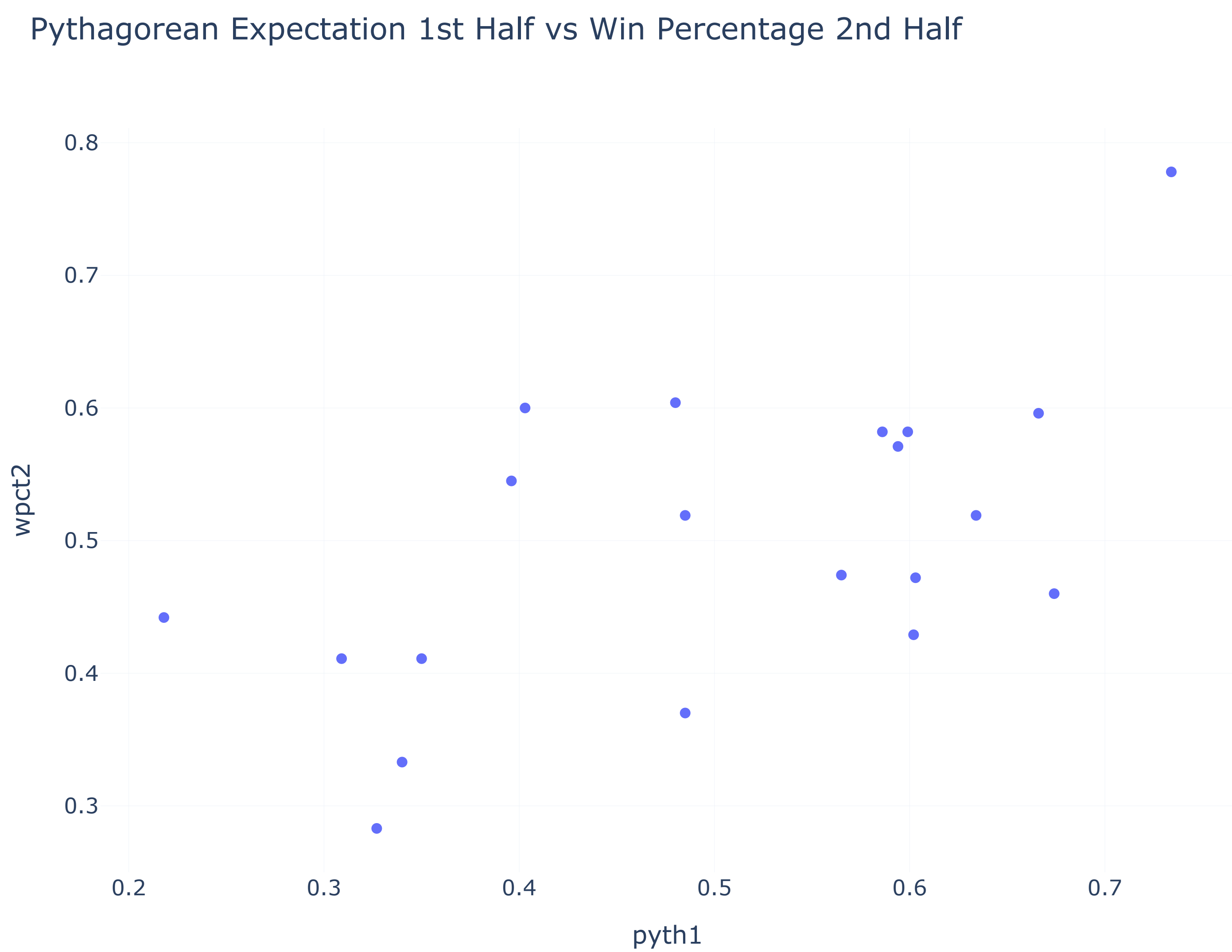
Calcular el porcentaje de victorias real y el pythagorean winning percentage para cada equipo en cada mitad de la temporada

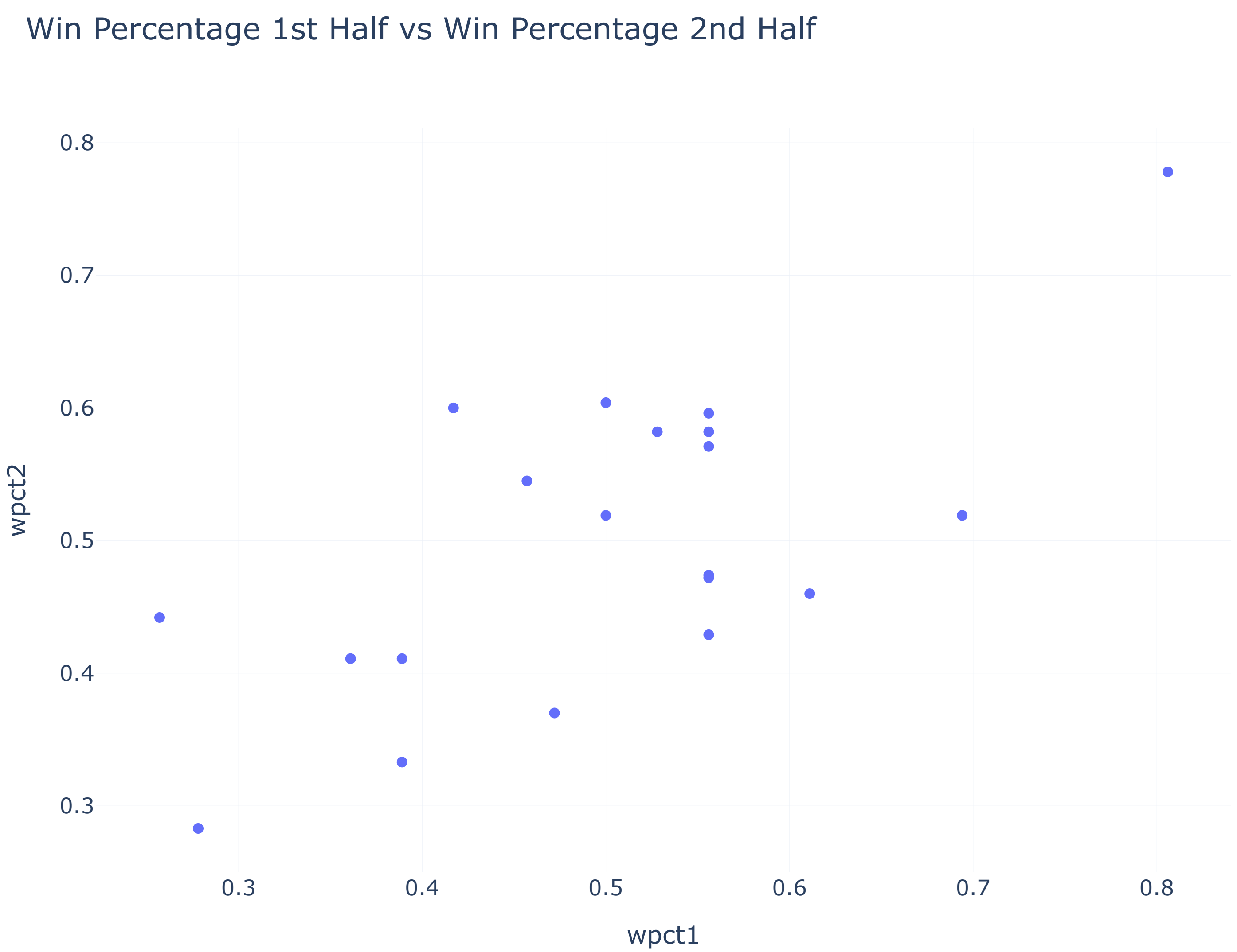
In [21]:
```
performance1stHalf['wpct1'] = round(performance1stHalf['win1'] / performance1stHalf['count1'], 3)
performance2ndHalf['wpct2'] = round(performance2ndHalf['win2'] / performance2ndHalf['count2'], 3)

performance1stHalf['pyth1'] = round((performance1stHalf['R1'] ** 2) / (performance1stHalf['R1'] ** 2 + perfor
mance1stHalf['RA1'] ** 2), 3)
performance2ndHalf['pyth2'] = round((performance2ndHalf['R2'] ** 2) / (performance2ndHalf['R2'] ** 2 + perfor
mance2ndHalf['RA2'] ** 2), 3)
```

Crear un df combinado con los datos de la primera y segunda mitad de la temporada para ver que tan bien pronostica el pythagorean expectation el rendimiento futuro de los equipos

In [22]:
```
predictor2ndHalf = pd.merge(performance1stHalf, performance2ndHalf, on='team')
```

In [23]:
```
fig1 = px.scatter(predictor2ndHalf, x='pyth1', y='wpct2', title='Pythagorean Expectation 1st Half vs Win Perc
entage 2nd Half')
fig1.update_layout(width=800, height=600, template='plotly_white')
fig1.show()
fig2 = px.scatter(predictor2ndHalf, x='wpct1', y='wpct2', title='Win Percentage 1st Half vs Win Percentage 2n
d Half')
fig2.update_layout(width=800, height=600, template='plotly_white')
fig2.show()
```


Pythagorean Expectation 1st Half vs Win Percentage 2nd Half
wpct2
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.3
0.4
0.5
0.6
0.7
pyth1




En la primer grafica los datos estan muy dispersos, por lo que puede parecer que no es una buena herramienta para predecir rendimientos futuros. En la segunda grafica se ve que tiene una distribucion muys similar a la primera por lo que podemos pensar que en alguna forma si esta coorelacionado el rendimiento futuro con el pythagorean expectation.

In [24]:
```
keyVariables = predictor2ndHalf[['wpct2', 'wpct1', 'pyth1', 'pyth2']].copy()
keyVariables.corr()
```

Out[24]:

| | wpct2 | wpct1 | pyth1 | pyth2 |
|---|---|---|---|---|
| **wpct2** | 1.000000 | 0.669649 | 0.583123 | 0.943724 |
| **wpct1** | 0.669649 | 1.000000 | 0.929732 | 0.648022 |
| **pyth1** | 0.583123 | 0.929732 | 1.000000 | 0.591563 |
| **pyth2** | 0.943724 | 0.648022 | 0.591563 | 1.000000 |

En esta matriz de correlacion vemos que para predecir el winning percentage de la segunda mitad es mejor hacerlo con el winning percentage de la primera mitad que con el pythagorean expectation de la primera mitad. Esto rompe con el supuesto de que el pythagorean expectation es una buena herramienta para predecir el rendimiento futuro de los equipos. Sin embargo la correlacion entre el pythagorean expectation de la primera mitad y el winning percentage de la segunda mitad es de 0.58 lo que indica que si tiene una relacion.

## Conclusiones

Esta herramienta es muy util para analizar el rendimiento actual de los equipos, en el caso de la LMB no es tan util para predecir el rendimiento futuro de los equipos. Se puede usar para ver si un equipo esta teniendo una temporada mejor o peor de lo esperado.

## References

[MLB-pythogorean winning percentage (https://www.mlb.com/glossary/advanced-stats/pythagorean-winning-percentage)](https://www.mlb.com/glossary/advanced-stats/pythagorean-winning-percentage)