Head coach dismissal effect on football team performance

Mathis Derenne[1], **Ewann x**[1], **Scott daz**[1], and **Romain dcv**[11]University of Rennes

Abstract

The goals of this paper is to investigate the effect of coach dismissal on team performance. To do that, we will use traditional statistical method that we apply to football teams. J'ajoute une ligne

1. Head coach dismissal effect on football team performance

1.a. Introduction

Sujet du TER : Comprendre l'effet du changement de club sur les performances du coach ET NON, comme le sujet initial (Rocaboy & Pavlik (2020)) Comprendre l'effet du changement de coach sur les performances du club Idée du prof : toutes choses étant égales par ailleurs (ceteris paribus) (idée d'un club représentatif), quelles sont les variations de performances d'un coach au cours du temps et lorsqu'il change de club

Impossible de créer "ce club égal par ailleurs" :

La création d'un club égal par ailleurs nécessite l'intervention d'un modèle qui permettrait, à partir des données du club (masse salariale, budget, performance passé du club, etc.) de normaliser la performance du club afin d'étudier précisemment l'impact du coach sur cette performance

Ceci pose plusieurs problèmes :

1. Les variations de performances du coach sont difficilement observable au travers la performance de l'équipe (détailler)
2. Impossible de respecter l'hypothèse d'uncounfoundness requise par de nombreux modèles statistiques corrigeant les externalités (ex: propensity score / PSM) (citer papier propensity score + expliquer l'idée du propensity score pour artificiellement recréer un groupe contrôle et test artificiel, expliquer l'hypothèse d'uncounfoundness et pourquoi elle est nécessaire)
3. Biais de causalité (point le plus important !) : on suppose que c'est la performance du coach qui fait varier la performance de l'équipe or, dès lors que cette causalité n'est plus vérifiée on se mord la queue dans la création du modèle explicatif :

Supposons que ce soit la performance de l'équipe qui causent les variations de performance du coach. Le modèle explicatif, censé créer ce club égale par ailleurs, va être amené à normaliser plus fortement un club qui performe bien par le passé. Or si c'est la performance de l'équipe qui cause la performance du coach on est en train de normaliser les variations de performance du coach. (mentionner l'existence de test d'inversion de la causalité + référence au papier) (expliquer ce que sont les fuites de données (data leakage) et que l'absence de cette hypothèse de causalité provoque des fuites de données entre les externalités et la variable d'interêt (la performance du coach)).

Published Feb 09, 2024

4. Le peu de donnée (retrouver le chiffre sur le nombre de club avec au moins 2 ou 3 changements de coach) (expliquer que dans la problématique initiale il y a bien plus de donnée car il y a davantage de club qui ont vu passer de coachs que de parcours individuel de coach au sein de clubs)
5. Problème de temporalité : les données sur les budgets des équipes, masse salariale ou valeur marchande des équipes ne sont pas disponible sous forme temporelle : impossible de savoir si la hausse de performance de l'équipe est dûe à la hausse du budget de l'équipe ou inversement.
6. Faible qualité des variables exogènes permettant l'analyse du système :

- Manque " d'objectivabilité " des variables externes : masse salariale (pas représentative, ex : sous-traitance), valeur marchande des joueurs (hautement subjectif)
- Manque de diversité des variables

Conclusion : Lors de l'analyse des effets dans un système, on raisonne généralement à petite entité égales par ailleurs Exemple : On parle d'agent économique représentatif et rarement d'une économie représentative :

- On observe l'effet de l'économie sur un agent économique
- et NON l'effet d'un agent économique sur l'économie

(à nuancer pour ne pas déplaire aux micro-économistes et rappeler le cadre statistiques de l'étude d'effets quantifiables !).

Référence à citer : https://clauswilke.com/dataviz/

2. Data extraction

2.a. Les données

- Utilisation de la librarie WorldFootBallR pour collecter des données
- les sites utilisés : Fbref et Transfermarkt (préciser le contenu pour les deux site)
- la fiabilité de ces sites et de ces données
- les données concernés

```
if (!require(worldfootballR)) {
  install.packages("worldfootballR")
}

if (!require(data.table)) {
  install.packages("data.table")
}

library(worldfootballR)
library(data.table)
```

```
Le chargement a n'ecessit'e le package : worldfootballR

Le chargement a n'ecessit'e le package : data.table
```

2.b. Get match results

```
country <- c("ENG", "ESP", "ITA", "GER", "FRA")
year <- c(2018, 2019, 2020, 2021, 2022)
result <- fb_match_results(country = country, gender = "M", season_end_year =
year, tier = "1st")
```

```
column_to_drop <- c('Gender', 'Day', 'Wk', 'Time', 'Venue', 'Referee',
'Attendance', 'Home_xG', 'Away_xG', 'Notes', 'Round')
result <- result[, !(names(result) %in% column_to_drop)]
# Rename result$Competition_Name of 'Fu\303\237ball-Bundesliga' to 'Bundesliga'
result$Competition_Name[result$Competition_Name == 'Fu\303\237ball-Bundesliga']
<- 'Bundesliga'
# Rename columns of result dataframe
names(result) <- c('league', 'country', 'season_year', 'date', 'home',
'home_goals', 'away', 'away_goals', 'match_url')
# Rename country code to country name
match_country <- c('ENG' = 'England', 'ITA' = 'Italy', 'FRA' = 'France', 'GER' =
'Germany', 'ESP' = 'Spain')
result$country <- match_country[result$country]
```

```
head(result)
paste(nrow(result), "rows")
print("leagues of interests:")
unique(result$league)
```

	league	country	season_year	date	home	home_goals	away	away_goals	match_url
	< chr>	< chr>	< int>	< date>	< chr>	< dbl>	< chr>	< dbl>	< chr>
1	Premier League	England	2018	2017-08-11	Arsenal	4	Leicester City	3	https://fbref.com/en/matches/e3c3ddf0/Arsenal-Leicester-City-August-11-2017-Premier-League
2	Premier League	England	2018	2017-08-12	Watford	3	Liverpool	3	https://fbref.com/en/matches/60f6cc1d/Watford-Liverpool-August-12-2017-Premier-League

	league	country	season_year	date	home	home_goals	away	away_goals	match_url
	< chr>	< chr>	< int>	< date>	< chr>	< dbl>	< chr>	< dbl>	< chr>
3	Premier League	England	2018	2017-08-12	Crystal Palace	0	Huddersfield	3	https://fbref.com/en/matches/2d369d17/Crystal-Palace-Huddersfield-Town-August-12-2017-Premier-League
4	Premier League	England	2018	2017-08-12	West Brom	1	Bournemouth	0	https://fbref.com/en/matches/684f704a/West-Bromwich-Albion-Bournemouth-August-12-2017-Premier-League
5	Premier League	England	2018	2017-08-12	Chelsea	2	Burnley	3	https://fbref.com/en/matches/71b00bca/Chelsea-Burnley-August-12-2017-Premier-League

	league	country	season_year	date	home	home_goals	away	away_goals	match_url
	< chr>	< chr>	< int>	< date>	< chr>	< dbl>	< chr>	< dbl>	< chr>
6	Premier League	England	2018	2017-08-12	Everton	1	Stoke City	0	https://fbref.com/en/matches/7c834541/Everton-Stoke-City-August-12-2017-Premier-League

'9148 rows'

```
[1] "leagues of interests:"
```

1. 'Premier League'
2. 'La Liga'
3. 'Ligue 1'
4. 'Bundesliga'
5. 'Serie A'

```
# Save result in 'data/match_results.csv'
fwrite(result, file = "data/extracted_match_results.csv", quote = "auto")
```

2.c. Get head Coach

```
country <- c("England", "Spain", "Italy", "Germany", "France")

# Créer le vecteur teams_url
teams_url <- c()
for (i in seq_along(country)) {
    team_url <- tm_league_team_urls(country_name = country[i], start_year =
2018)
    print(paste(country[i], ":", length(team_url), "teams"))
    teams_url <- c(teams_url, team_url)
}
```

```
[1] "England : 20 teams"
[1] "Spain : 20 teams"
[1] "Italy : 20 teams"
[1] "Germany : 18 teams"
[1] "France : 20 teams"
```

```
head_coach <- tm_team_staff_history(team_urls = teams_url, staff_role =
"Manager")
print(paste(nrow(head_coach), "head coaches records"))
```

```
[1] "4855 head coaches records"
```

```
# match_results$league : 'Premier League''La Liga''Ligue 1''Bundesliga''Serie A'
unique(head_coach$league)
league <- c('Premier League', 'LaLiga', 'Ligue 1', 'Bundesliga', 'Serie A')
head_coach_bis <- head_coach[head_coach$league %in% league,]
paste(nrow(head_coach_bis), "head coaches records for leagues of interests")
```

1. 'Premier League'
2. 'Championship'
3. 'LaLiga'
4. 'LaLiga2'
5. 'Serie A'
6. 'Serie B'
7. 'Serie C - Girone B'
8. NA
9. 'Bundesliga'
10. '2. Bundesliga'
11. 'Ligue 1'
12. 'Ligue 2'
13. 'Championnat National'

'3528 head coaches records for leagues of interests'

Leagues in which we are collecting data :

head coaches records for leagues of interests

```
# Select head-coach that have been active between 2018 and 2022
head_coach_bis <- head_coach_bis[is.na(head_coach_bis$end_date) |
head_coach_bis$end_date >= "2018-01-01",]
head_coach_bis <- head_coach_bis[head_coach_bis$appointed <= "2022-12-31",]
paste(nrow(head_coach_bis), "head coaches records for leagues of interests
active between 2018 and 2022")
# Drop column
column_to_drop <- c("staff_role", "ppg")
head_coach_bis <- head_coach_bis[, !(names(head_coach_bis) %in% column_to_drop)]
# Rename staff_name column to coach_name
names(head_coach_bis)[names(head_coach_bis) == "staff_name"] <- "coach_name"
```

'298 head coaches records for leagues of interests active between 2018 and 2022'

```
# Save in 'data/head_coach.csv'
fwrite(head_coach_bis, file = "data/extracted_head_coach.csv", quote = "auto")
```

3. Preprocessing

```
import pandas as pd

match_results = pd.read_csv('data/extracted_match_results.csv',
parse_dates=['date'])
head_coach = pd.read_csv('data/extracted_head_coach.csv',
parse_dates=['appointed', 'end_date'])

match_results.drop(columns = ['match_url'], inplace = True)
match_results.rename(columns = {'home': 'home_team', 'away': 'away_team'},
inplace = True)
head_coach.drop(columns = ['staff_url'], inplace = True)
```

```
head_coach.rename(columns = {'team_name': 'team'}, inplace = True)

display(match_results.head())
display(head_coach.head())
```

	league	country	season_year	date	home_team	home_goals	away_team	away_goals
0	Premier League	England	2018	2017-08-11	Arsenal	4.0	Leicester City	3.0
1	Premier League	England	2018	2017-08-12	Watford	3.0	Liverpool	3.0
2	Premier League	England	2018	2017-08-12	Crystal Palace	0.0	Huddersfield	3.0
3	Premier League	England	2018	2017-08-12	West Brom	1.0	Bournemouth	0.0
4	Premier League	England	2018	2017-08-12	Chelsea	2.0	Burnley	3.0

	team	league	country	coach_name	staff_dob	staff_nat	staff_mat	appointed_date	end_date	days_in_charge	matches	wins	draws	losses
0	Manchester City	Premier League	England	Pep Guardiola	Jan 18, 1971	Spain	NaN	2016-07-01	NaT	2784	450	333	53	64
1	Liverpool FC	Premier League	England	Jürgen Klopp	Jun 16, 1967	Germany	NaN	2015-10-08	2024-01-28	3130	468	291	96	81
2	Chelsea FC	Premier League	England	Graham Potter	May 20, 1975	England	NaN	2022-09-08	2023-04-02	206	31	12	8	11
3	Chelsea FC	Premier League	England	Thomas Tuchel	Aug 29, 1973	Germany	NaN	2021-01-26	2022-09-07	589	100	63	19	18
4	Chelsea FC	Premier League	England	Frank Lampard	Jun 20, 1978	England	NaN	2019-07-01	2021-01-25	573	84	44	15	25

3.a. Team's name

```
# Compute number of team that are in head_coach but not in match_results
coach_team = set(head_coach['team'])
match_team = set(match_results['home_team']) | set(match_results['away_team'])
coach_team_not_in_match = coach_team - match_team
match_team_not_in_coach = match_team - coach_team

len(coach_team_not_in_match), len(match_team_not_in_coach)
```

```
(63, 132)
```

In total, match_results dataset contains teams and head_coach dataset contains teams. However some teams name are different between the two datasets. For example 'Liverpool' in match_results is 'Liverpool FC' in head_coach. This is problematic as we will need to join data on team's columns.

In total there is teams present in match_results but not in head_coach and teams present in head_coach but not in match_results. It indicates that despite mismatched names, that there are several teams present in match_results which do not have records of a coach. (needs more explaination in Data Extraction about data and why this is surprising based on how we filter head coach to at least include latest head coach).

Addressing this surprise ...

To address mismatched teams name we will use Levenshtein Distance (add reference to paper) to match team's name of head_coach missing in match teams with match teams.

```
from thefuzz import process

def match_names(name, list_names, min_score=0):
    scores = process.extract(name, list_names, limit=1)

    if len(scores) > 0 and scores[0][1] >= min_score:
        return scores[0][0]
    return None

name_match = {}

for team in coach_team:
    match = match_names(team, match_team, min_score=60)
    if match is not None:
        name_match[team] = match
    else:
        name_match[team] = None
        print(f"No match found for {team}")

# Show name_match
for team, match in name_match.items():
    print(f"{team:30} matched with {match}")
```

```
Arsenal FC                     matched with Arsenal
FC Nantes                      matched with Nantes
Frosinone Calcio               matched with Frosinone
Rayo Vallecano                 matched with Rayo Vallecano
Stade Reims                    matched with Reims
SS Lazio                       matched with Lazio
Inter Milan                    matched with Inter
Brighton & Hove Albion         matched with Brighton
Sevilla FC                     matched with Sevilla
RB Leipzig                     matched with RB Leipzig
Borussia Mönchengladbach       matched with M'Gladbach
FC Augsburg                    matched with Augsburg
OGC Nice                       matched with Nice
Genoa CFC                      matched with Genoa
Chelsea FC                     matched with Chelsea
Deportivo Alavés               matched with Alavés
Newcastle United               matched with Newcastle Utd
Manchester United              matched with Manchester Utd
Stade Rennais FC               matched with Rennes
SSC Napoli                     matched with Napoli
VfL Wolfsburg                  matched with Wolfsburg
Montpellier HSC                matched with Montpellier
TSG 1899 Hoffenheim            matched with Hoffenheim
```

```
AS Roma                         matched with Roma
Celta de Vigo                   matched with Celta Vigo
Torino FC                       matched with Torino
US Sassuolo                     matched with Sassuolo
Burnley FC                      matched with Burnley
Olympique Marseille             matched with Marseille
1.FSV Mainz 05                  matched with Mainz 05
Bayern Munich                   matched with Bayern Munich
SV Werder Bremen                matched with Werder Bremen
SC Freiburg                     matched with Freiburg
Real Sociedad                   matched with Real Sociedad
Olympique Lyon                  matched with Lyon
Real Betis Balompié             matched with Betis
West Ham United                 matched with West Ham
Athletic Bilbao                 matched with Athletic Club
Fulham FC                       matched with Fulham
Borussia Dortmund               matched with Dortmund
AFC Bournemouth                 matched with Bournemouth
Paris Saint-Germain             matched with Paris S-G
Atlético de Madrid              matched with Atlético Madrid
VfB Stuttgart                   matched with Stuttgart
AS Monaco                       matched with Monaco
Tottenham Hotspur               matched with Tottenham
Getafe CF                       matched with Getafe
Bologna FC 1909                 matched with Bologna
Eintracht Frankfurt             matched with Eint Frankfurt
RC Strasbourg Alsace            matched with Strasbourg
Valencia CF                     matched with Valencia
Villarreal CF                   matched with Villarreal
AC Milan                        matched with Milan
Udinese Calcio                  matched with Udinese
ACF Fiorentina                  matched with Fiorentina
Bayer 04 Leverkusen             matched with Leverkusen
LOSC Lille                      matched with Lille
Girona FC                       matched with Girona
Manchester City                 matched with Manchester City
Cagliari Calcio                 matched with Cagliari
FC Barcelona                    matched with Barcelona
FC Empoli                       matched with Empoli
Crystal Palace                  matched with Crystal Palace
FC Toulouse                     matched with Toulouse
Everton FC                      matched with Everton
Liverpool FC                    matched with Liverpool
Wolverhampton Wanderers         matched with Wolves
Real Madrid                     matched with Real Madrid
Juventus FC                     matched with Juventus
Atalanta BC                     matched with Atalanta
```

```python
# # Fix some names
# name_match['Inter Milan'] = 'Inter'
# name_match['AC Milan'] = 'Milan'
# name_match['Stade Rennais FC'] = 'Rennes'

# Ensure everything map
for team in coach_team:
    if name_match[team] is None:
        print(f"No match found for {team}")
```

```
# Map head_coach['team'] with name_match
head_coach['team'] = head_coach['team'].map(name_match)
```

3.b. To-Do

- investigate NaN values
- investigate inf and -inf values

3.c. Saving preprocessed data

```
# Save match_results
match_results.to_csv('data/match_results.csv', index=False)
head_coach.to_csv('data/head_coach.csv', index=False)
```

4. Head coach Exploratory data analysis

4.a. Imports

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
import matplotlib.colors as mcolors
import matplotlib.cm as cm
import seaborn as sns

from scipy.stats import pearsonr, pointbiserialr
from IPython.display import display, Markdown, HTML
from datetime import datetime

sns.set_style()
sns.set_theme(style = 'ticks', palette = 'pastel')
plt.rcParams['figure.autolayout'] = True
plt.rcParams['savefig.bbox'] = 'tight'
sns.set_context("paper")

plt.rcParams['savefig.dpi'] = 300
plt.rcParams['savefig.bbox'] = 'tight'
plt.rcParams['savefig.directory'] = 'figures'
```

4.b. Loading data

```
head_coach = pd.read_csv('data/head_coach.csv', parse_dates=['appointed',
'end_date'])

long_tenure = head_coach[head_coach['days_in_post'] > 3000].shape[0]
print(f"We will exclude head coaches with more than 3000 days in post. There are
{long_tenure} head coaches with more than 3000 days in post.")
# Explain that those are outliers
head_coach = head_coach[head_coach['days_in_post'] <= 3000]
display(head_coach.head())
```

```
We will exclude head coaches with more than 3000 days in post. There are 5 head
coaches with more than 3000 days in post.
```

	team	league	country	coach	staff_id	staff_nat	nstaff_nat	appointed	end_date	days_in_post	matches	wins	draws	losses
0	Manchester City	Premier League	England	Pep Guardiola	Jan 18, 1971	Spain	NaN	2016-07-01	NaT	2784	450	333	53	64

	team	league	country	coach_staff	staff_dob	staff_nat	staff_appointed	end_date	days_in_post	games	wins	draws	losses
2	Chelsea	Premier League	England	Graham Potter	May 20, 1975	England	NaN	2022-09-08	2023-04-02	206	31	12	8	11
3	Chelsea	Premier League	England	Thomas Tuchel	Aug 29, 1973	Germany	NaN	2021-01-26	2022-09-07	589	100	63	19	18
4	Chelsea	Premier League	England	Frank Lampard	Jun 20, 1978	England	NaN	2019-07-01	2021-01-25	573	84	44	15	25
5	Chelsea	Premier League	England	Maurizio Sarri	Jan 10, 1959	Italy	NaN	2018-07-01	2019-07-30	393	63	40	11	12

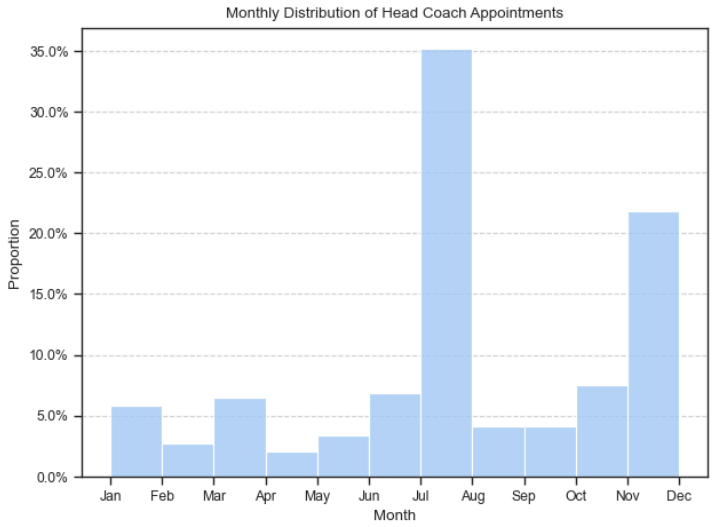
4.c. General information about data

Data collected from match results ranges from to and contains the results of matches.

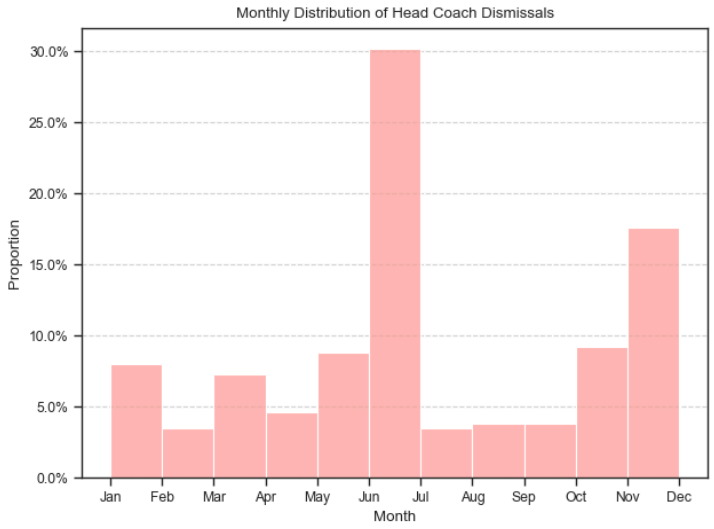
Matches have been collected for the following leagues :

4.d. Basic plots

```
# Useful to add xtick months to dayofyear plot
months = pd.date_range('2022-01-01', '2022-12-31',
freq='ME').strftime('%b').tolist()

# Plot for Head Coach appointed distribution
plt.figure()
sns.histplot(head_coach['appointed'].dt.month, bins=11, color = 'b', kde=False,
stat='density')
plt.gca().yaxis.set_major_formatter(ticker.PercentFormatter(xmax=1))
plt.xticks(range(1, 13), months)
plt.title('Monthly Distribution of Head Coach Appointments')
plt.xlabel('Month')
plt.ylabel('Proportion')
plt.grid(axis='y', linestyle='--', alpha=0.8)
plt.savefig('figures/hc_appointment.png')
```

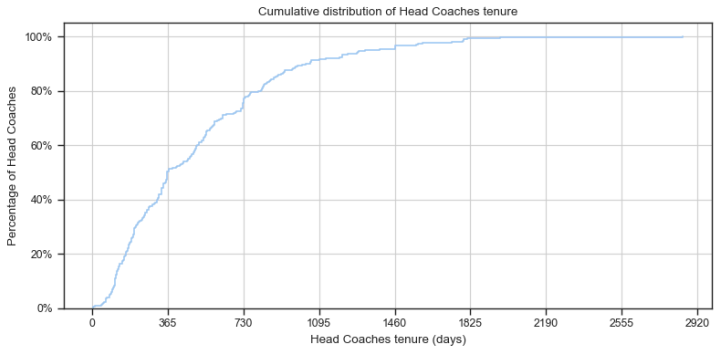



```
# Plot for Head Coach dismissal distribution
plt.figure()
sns.histplot(head_coach['end_date'].dt.month, bins=11, color = 'r', kde=False,
stat='density')
plt.xticks(range(1, 13), months)
plt.gca().yaxis.set_major_formatter(ticker.PercentFormatter(xmax=1))
plt.title('Monthly Distribution of Head Coach Dismissals')
plt.xlabel('Month')
plt.ylabel('Proportion')
plt.grid(axis='y', linestyle='--', alpha=0.8)
plt.savefig('figures/hc_dismissal.png')
```



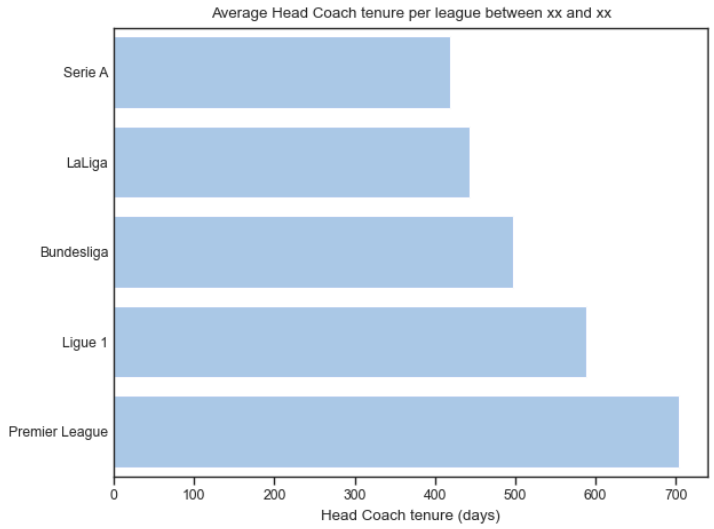

```
head_coach_dismissed = head_coach[head_coach['end_date'].notnull()]

# Plot ECDF of head_coach['days_in_post']
plt.figure(figsize=(8, 4))

sns.ecdfplot(data=head_coach_dismissed, x='days_in_post', stat = 'percent')
plt.ylabel('Percentage of Head Coaches')
# Format percentage
```

```
plt.gca().yaxis.set_major_formatter(ticker.PercentFormatter(xmax=100))
# Grid
plt.grid(axis='y', linestyle='-', alpha=0.8)
plt.grid(axis='x', linestyle='-', alpha=0.8)
plt.xticks(range(0, max(head_coach_dismissed['days_in_post']) + 365, 365))
# plt.xlim(0, head_coach_dismissed['days_in_post'].quantile(0.97))
plt.title('Cumulative distribution of Head Coaches tenure')
plt.xlabel('Head Coaches tenure (days)')
plt.savefig('figures/hc_tenure.png')
```

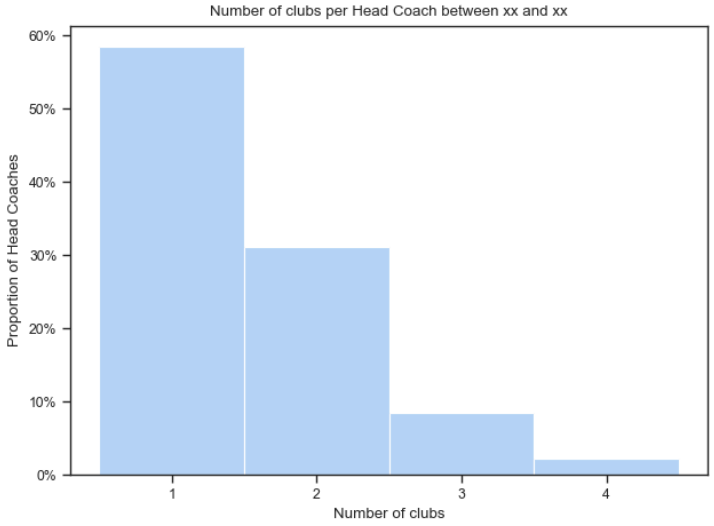


En moyenne, les coachs sont restés en poste jours.

```
# Average days in post per league

# Calculate average days in post per league
avg_days_in_post = head_coach_dismissed.groupby('league')['days_in_post'].mean()
avg_days_in_post = avg_days_in_post.sort_values()

# Plot average days in post per league
plt.figure()
sns.barplot(y=avg_days_in_post.index, x=avg_days_in_post.values, orient='h')
plt.title('Average Head Coach tenure per league between xx and xx')
plt.xlabel('Head Coach tenure (days)')
plt.tick_params(axis='y', which='both', length=0)
# Disable ylabel
plt.ylabel('')

plt.savefig('figures/hc_tenure_per_league.png')
```

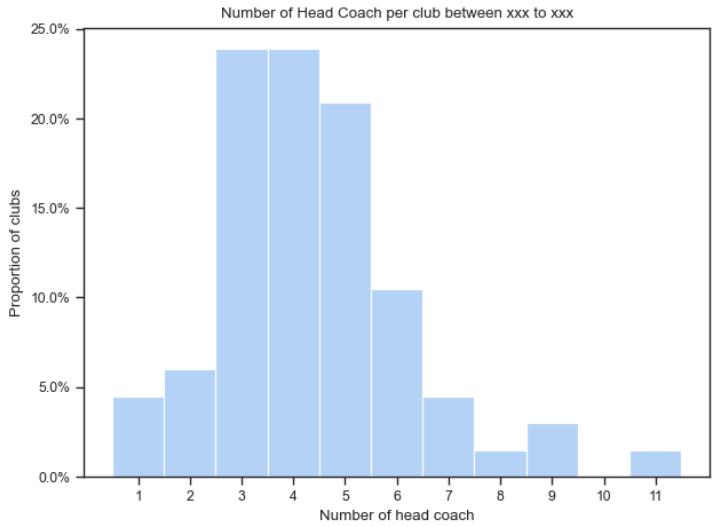



```
# Number of clubs per Head Coach

# Group by coach_name and count the number of clubs
club_per_coach =
head_coach.groupby('coach_name').size().reset_index(name='count')

plt.figure()
sns.histplot(x='count', data = club_per_coach, discrete = True,
stat="probability")
plt.xticks(range(1, club_per_coach['count'].max() + 1))
plt.gca().yaxis.set_major_formatter(ticker.PercentFormatter(xmax=1))
plt.title('Number of clubs per Head Coach between xx and xx')
plt.xlabel('Number of clubs')
plt.ylabel('Proportion of Head Coaches')
plt.savefig('figures/number_of_club_per_coach.png')
```

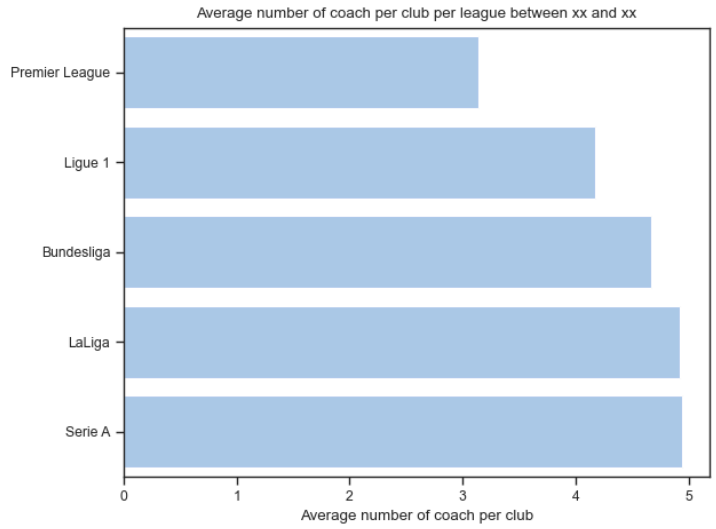



```
# Number of Head Coachs per club

# Group by team and count the number of head coach
coach_per_club = head_coach.groupby('team').size().reset_index(name='count')
```

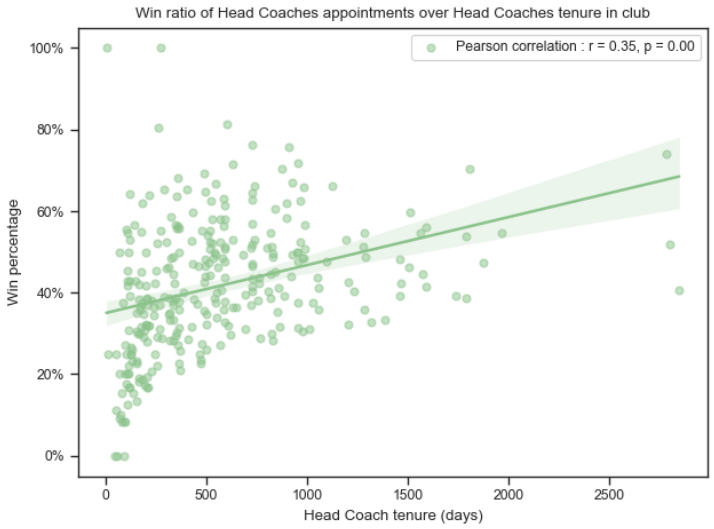
```
plt.figure()
sns.histplot(x='count', data = coach_per_club, discrete=True,
stat="probability")
plt.xticks(range(1, coach_per_club['count'].max() + 1))
plt.gca().yaxis.set_major_formatter(ticker.PercentFormatter(xmax=1))
plt.title(f'Number of Head Coach per club between xxx to xxx')
plt.xlabel('Number of head coach')
plt.ylabel('Proportion of clubs')
plt.savefig('figures/number_of_coach_per_club.png')
```

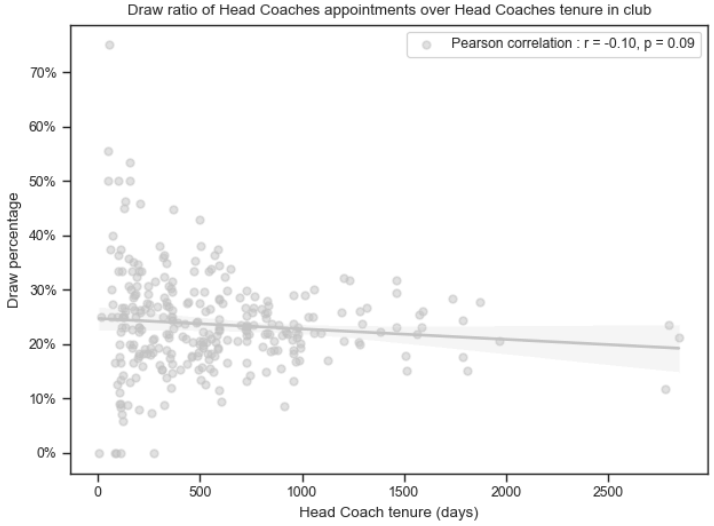


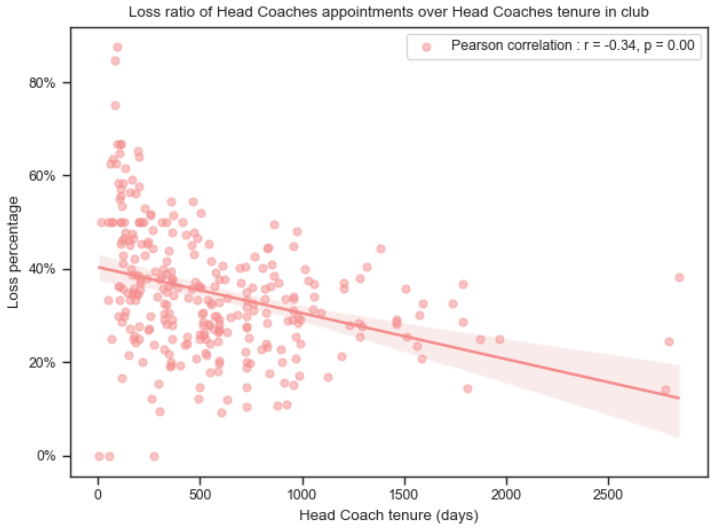

```
# Average number of coach per club per league

# Calculate average number of coach per club per league
coach_per_team = head_coach.groupby(['league', 'team']).size()
avg_number_of_coach_per_club_per_league =
coach_per_team.groupby('league').mean().sort_values()

# Plot average number of coach per club per league
plt.figure()
sns.barplot(x=avg_number_of_coach_per_club_per_league.values,
y=avg_number_of_coach_per_club_per_league.index, orient='h')
plt.title('Average number of coach per club per league between xx and xx')
plt.ylabel('')
plt.xlabel('Average number of coach per club')
plt.savefig('figures/hc_per_club_per_league.png')
```



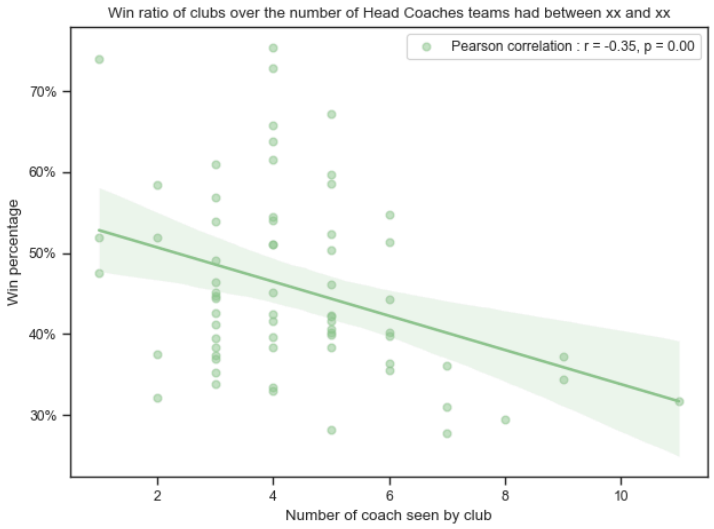

```
# Plot of wins, draw and losses percentage over days in post
head_coach['win_percentage'] = head_coach['wins'] / head_coach['matches'] * 100
head_coach['draw_percentage'] = head_coach['draws'] / head_coach['matches'] *
100
head_coach['loss_percentage'] = head_coach['losses'] / head_coach['matches'] *
100

def plot_percentage_over_days(data, y_value, y_leg, color):
    plt.figure()
    sns.regplot(x='days_in_post', y=y_value, data=data,
color=sns.light_palette(color, as_cmap=True)(0.4), scatter_kws={'alpha':0.5},
label = y_leg + ' ratio')

    plt.title(f'{y_leg} ratio of Head Coaches appointments over Head Coaches
tenure in club')
    ## x = head coach tenure in a club
    ## y = win/draw/loss ratio under the appointment of the head coach
    plt.xlabel('Head Coach tenure (days)')
    plt.ylabel(f'{y_leg.capitalize()} percentage')
    plt.gca().yaxis.set_major_formatter(ticker.PercentFormatter(xmax=100))

    # Calculate Pearson correlation coefficient
    r, p = pearsonr(data['days_in_post'], data[y_value])
    plt.legend([f'Pearson correlation : r = {r:.2f}, p = {p:.2f}'], loc='upper
right')

    plt.savefig(f'figures/{y_value}_over_hc_tenure.png')

plot_percentage_over_days(head_coach, 'win_percentage', 'Win', 'green')
plot_percentage_over_days(head_coach, 'draw_percentage', 'Draw', 'gray')
plot_percentage_over_days(head_coach, 'loss_percentage', 'Loss', 'red')
```

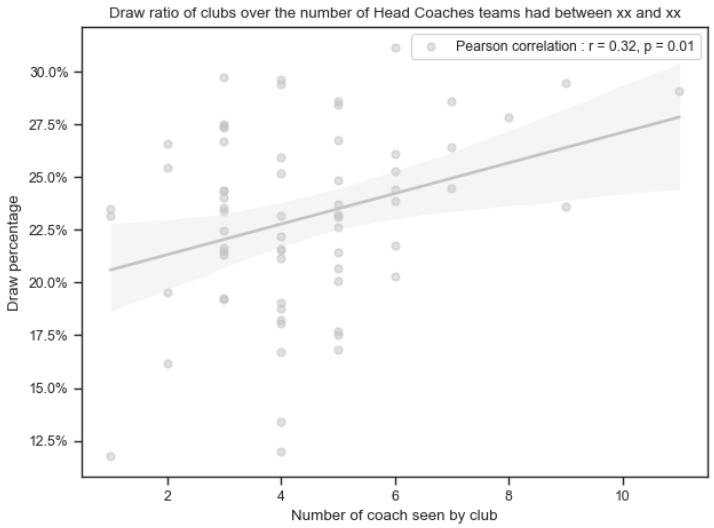

Win ratio of Head Coaches appointments over Head Coaches tenure in club
Pearson correlation : r = 0.35, p = 0.00
Win percentage
100%
80%
60%
40%
20%
0%
0
500
1000
1500
2000
2500
Head Coach tenure (days)

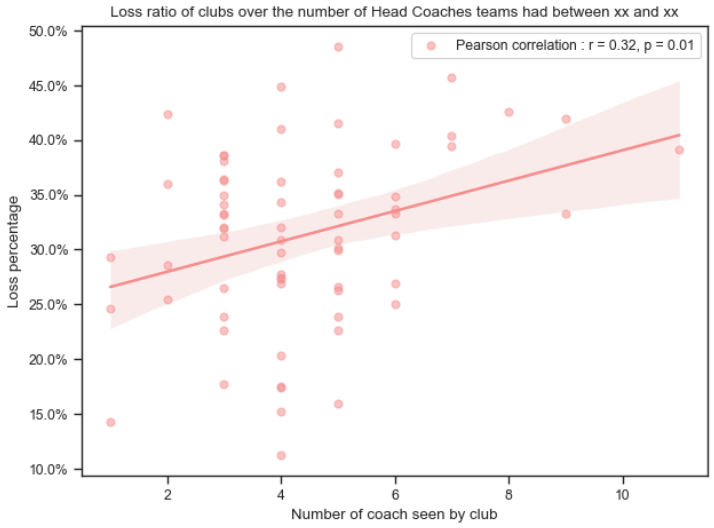

Draw ratio of Head Coaches appointments over Head Coaches tenure in club
Pearson correlation : r = -0.10, p = 0.09
Draw percentage
70%
60%
50%
40%
30%
20%
10%
0%
0
500
1000
1500
2000
2500
Head Coach tenure (days)


Loss ratio of Head Coaches appointments over Head Coaches tenure in club
Pearson correlation : r = -0.34, p = 0.00
Loss percentage
80%
60%
40%
20%
0%
0
500
1000
1500
2000
2500
Head Coach tenure (days)

```
# Plot match outcome of clubs over number of coach seen by club

club_results = head_coach.groupby('team').agg({'wins': 'sum', 'draws': 'sum',
'losses': 'sum', 'matches': 'sum', 'coach_name': 'count'})
club_results = club_results.rename(columns={'coach_name': 'coach_count'})
club_results['win_percentage'] = club_results['wins'] / club_results['matches']
* 100
club_results['draw_percentage'] = club_results['draws'] /
club_results['matches'] * 100
club_results['loss_percentage'] = club_results['losses'] /
club_results['matches'] * 100

# Plot percentage over number of coach seen by club
def plot_club_outcome(data, y_value, y_leg, color):
    plt.figure()
    sns.regplot(x='coach_count', y=y_value, data=data,
color=sns.light_palette(color, as_cmap=True)(0.4), scatter_kws={'alpha':0.5},
label = y_leg + ' ratio')
    # x = number of coach seen by club
    # y = win/draw/loss ratio of the club
    plt.title(f'{y_leg} ratio of clubs over the number of Head Coaches teams had
between xx and xx')
    plt.xlabel('Number of coach seen by club')
    plt.ylabel(f'{y_leg} percentage')
    plt.gca().yaxis.set_major_formatter(ticker.PercentFormatter(xmax=100))

    r, p = pearsonr(data['coach_count'], data[y_value])
    plt.legend([f'Pearson correlation : r = {r:.2f}, p = {p:.2f}'], loc='upper
right')

    plt.savefig(f'figures/{y_value}_over_club_hc_count.png')

plot_club_outcome(club_results, 'win_percentage', 'Win', 'green')
plot_club_outcome(club_results, 'draw_percentage', 'Draw', 'gray')
plot_club_outcome(club_results, 'loss_percentage', 'Loss', 'red')
```

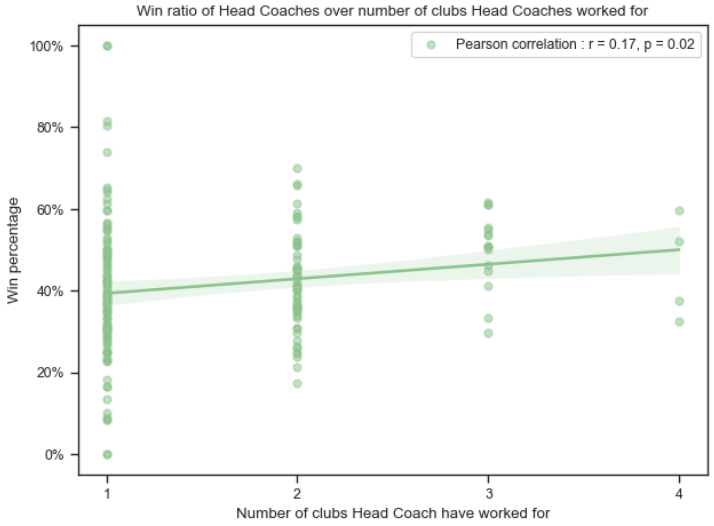


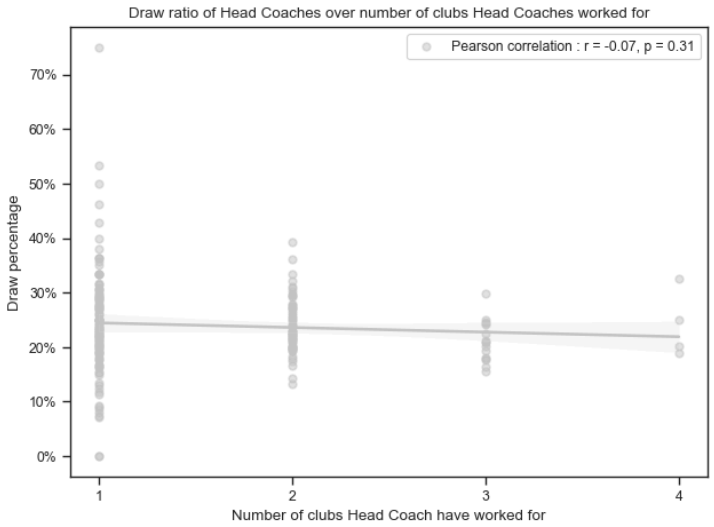


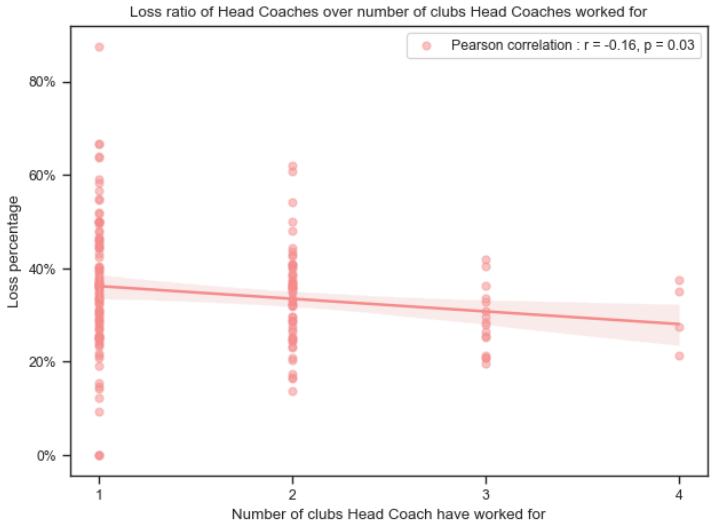



```
# Plot of wins, draw and losses percentage over number of club head coach has
been

head_coach_performance = head_coach.groupby('coach_name').agg({'matches': 'sum',
'wins': 'sum', 'draws': 'sum', 'losses': 'sum', 'team': 'count'}).reset_index()
head_coach_performance = head_coach_performance.rename(columns={'team':
'club_count'})
head_coach_performance['win_percentage'] = head_coach_performance['wins'] /
head_coach_performance['matches'] * 100
head_coach_performance['draw_percentage'] = head_coach_performance['draws'] /
head_coach_performance['matches'] * 100
head_coach_performance['loss_percentage'] = head_coach_performance['losses'] /
head_coach_performance['matches'] * 100

# Linear regression plot for wins, draw and losses percentage over number of
club head coach has been

def plot_percentage_over_club_count(data, y_value, y_leg, color):
    plt.figure()
```

```
    sns.regplot(x='club_count', y=y_value, data=data, color =
sns.light_palette(color, as_cmap=True)(0.4), scatter_kws={'alpha':0.5}, label =
y_leg + ' ratio')
    # x = number of club head coach has worked for
    # y = win/draw/loss ratio of the head coach over all clubs
    plt.title(f'{y_leg.capitalize()} ratio of Head Coaches over number of clubs
Head Coaches worked for')
    plt.xticks(range(1, data['club_count'].max() + 1))
    plt.xlabel('Number of clubs Head Coach have worked for')
    plt.ylabel(f'{y_leg.capitalize()} percentage')
    plt.gca().yaxis.set_major_formatter(ticker.PercentFormatter(xmax=100))

    r, p = pearsonr(data['club_count'], data[y_value])
    plt.legend([f'Pearson correlation : r = {r:.2f}, p = {p:.2f}'], loc='upper
right')

    plt.savefig(f'figures/{y_value}_over_club_per_hc_count.png')

plot_percentage_over_club_count(head_coach_performance, 'win_percentage', 'win',
'green')
plot_percentage_over_club_count(head_coach_performance, 'draw_percentage',
'draw', 'gray')
plot_percentage_over_club_count(head_coach_performance, 'loss_percentage',
'loss', 'red')
```







5. Match results exploratory data analysis

5.a. Imports

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
import matplotlib.colors as mcolors
import matplotlib.cm as cm
import seaborn as sns
from IPython.display import display, Markdown, HTML
from datetime import datetime

sns.set_theme(style = 'ticks', palette = 'pastel')
plt.rcParams['figure.autolayout'] = True
plt.rcParams['savefig.bbox'] = 'tight'
sns.set_context("paper")
```

```
# Define fig saving context
plt.rcParams['savefig.dpi'] = 300
plt.rcParams['savefig.bbox'] = 'tight'
plt.rcParams['savefig.directory'] = 'figures'


from utils.utils import filter_team, league_team, team_league, unique_teams,
unique_teams_coach_change
# league_team : league -> [team]
# team_league : team -> league
# unique_teams : [all teams]
# unique_teams_coach_change : [all teams that have had a coach change]
```

5.b. Loading data

```
match_results = pd.read_csv('data/match_results.csv', parse_dates=['date'])
# head_coach = pd.read_csv('data/head_coach.csv', parse_dates=['appointed',
'end_date'])

match_results.head()
```

	league	country	season_year	date	home_team	home_goals	away_team	away_goals
0	Premier League	England	2018	2017-08-11	Arsenal	4.0	Leicester City	3.0
1	Premier League	England	2018	2017-08-12	Watford	3.0	Liverpool	3.0
2	Premier League	England	2018	2017-08-12	Crystal Palace	0.0	Huddersfield	3.0
3	Premier League	England	2018	2017-08-12	West Brom	1.0	Bournemouth	0.0
4	Premier League	England	2018	2017-08-12	Chelsea	2.0	Burnley	3.0

```
# General information
min_year = match_results['date'].min()
max_year = match_results['date'].max()
```

5.c. General information about data

Data collected from match results ranges from to and contains the results of matches.

Matches have been collected for the following leagues :

```
# Leagues informations
match_results['total_goals'] = match_results['home_goals'] +
match_results['away_goals']
leagues = (match_results.groupby(['league', 'country'])
            .agg(matches_played=('home_team', 'count'),
avg_goals=('total_goals', 'mean'))
            .reset_index()
            .assign(league_country=lambda df: df['league'] + ' (' +
df['country'] + ')')
            .sort_values(by='matches_played', ascending=False))
# Number of teams : use league_team = {league: [team]} to get the number of
teams in each league and add it to league dataframe
leagues['number_of_teams'] = leagues['league'].apply(lambda x:
```

```
len(league_team[x]))

league_team_coach_change = {league: [team for team in league_team[league] if
team in unique_teams_coach_change] for league in league_team}
leagues['number_of_teams_with_coach_change'] = leagues['league'].apply(lambda x:
len(league_team_coach_change[x]))
# Round avg goals
leagues['avg_goals'] = leagues['avg_goals'].round(2)
# Drop league and country columns
leagues = leagues.drop(columns=['league', 'country'])
leagues = leagues.set_index('league_country')
# Rename columns with proper formatting
leagues = leagues.rename(columns={'matches_played': 'Number of match played',
'avg_goals': 'Average goals', 'number_of_teams': 'Number of teams',
'number_of_teams_with_coach_change': 'Number of teams with coach change'})
# Reorder columns
leagues = leagues[['Number of match played', 'Average goals', 'Number of teams',
'Number of teams with coach change']]
# Rename index
leagues.index.name = 'Leagues'
display(HTML(leagues.to_html()))
```

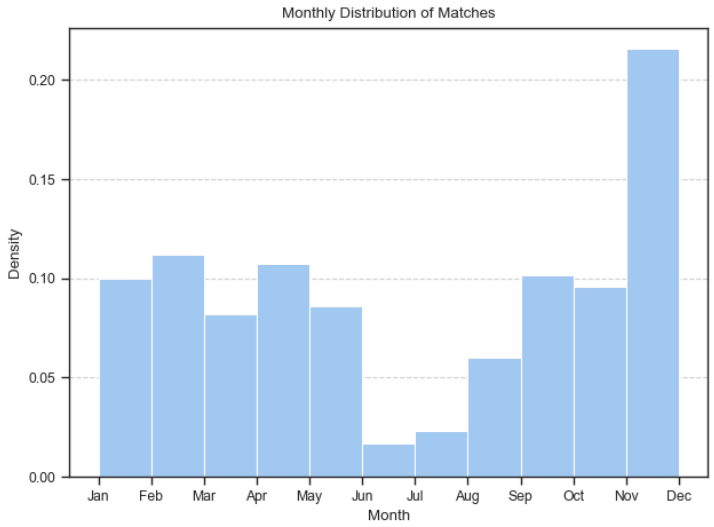
	Number of match played	Average goals	Number of teams	Number of teams with coach change
Leagues				
Ligue 1 (France)	1908	2.68	28	12
La Liga (Spain)	1900	2.55	28	14
Premier League (England)	1900	2.75	28	15
Serie A (Italy)	1900	2.86	28	16
Bundesliga (Germany)	1540	3.06	27	13

```
for team in unique_teams:
    if filter_team(team).shape[0] !=
filter_team(team).drop_duplicates().shape[0]:
        print(f"Team {team} has more than one match in the same day")
# No team played more than one match in the same day (ie. (date, team) can be
index of match_result)
```

5.d. Basic plots

```
# Useful to add xtick months to dayofyear plot
months = pd.date_range('2022-01-01', '2022-12-31',
freq='ME').strftime('%b').tolist()
days = np.linspace(1, 365, num=12, dtype=int)

# Plot distribution of match
plt.figure()
plt.hist(match_results['date'].dt.month, bins=11, color = 'b', density=True)
plt.title('Monthly Distribution of Matches')
plt.xlabel('Month')
plt.ylabel('Density')
plt.xticks(range(1, 13), months)
```

```
plt.grid(axis='y', linestyle='--', alpha=0.8)
plt.savefig('figures/match_distribution.png')
```




```
match_results.head()
```

	league	country	season_year	date	home_team	home_goals	away_team	away_goals	total_goals
0	Premier League	England	2018	2017-08-11	Arsenal	4.0	Leicester City	3.0	7.0
1	Premier League	England	2018	2017-08-12	Watford	3.0	Liverpool	3.0	6.0
2	Premier League	England	2018	2017-08-12	Crystal Palace	0.0	Huddersfield	3.0	3.0
3	Premier League	England	2018	2017-08-12	West Brom	1.0	Bournemouth	0.0	1.0
4	Premier League	England	2018	2017-08-12	Chelsea	2.0	Burnley	3.0	5.0

```
total_matches = match_results.shape[0]
home_goals = match_results['home_goals'].mean()
away_goals = match_results['away_goals'].mean()
diff_goal_perc = ((home_goals - away_goals) / away_goals) * 100

home_win = (match_results['home_goals'] > match_results['away_goals']).sum()
away_win = (match_results['home_goals'] < match_results['away_goals']).sum()
diff_win_perc = ((home_win - away_win) / away_win) * 100

draw_count = (match_results['home_goals'] == match_results['away_goals']).sum()
draw_perc = (draw_count / (home_win + away_win)) * 100
```

5.d.i. *Venue effect on team's performance:*

Il existe une différence dans la performance des équipes lorsqu'elle joue à domicile ou à l'extérieur (voir Figure 1).

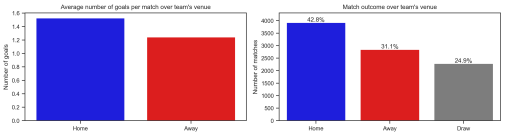
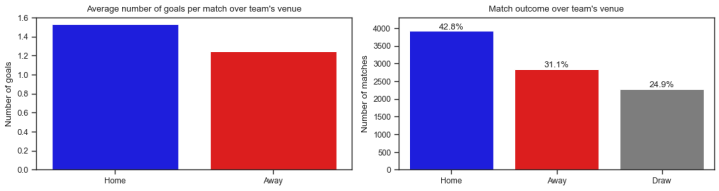


Figure 1: Venue effect on team's performance

```
fig, ax = plt.subplots(1, 2, figsize=(11, 3))

sns.barplot(x=['Home', 'Away'], y=[home_goals, away_goals], ax=ax[0],
palette=['blue', 'red'], hue = ['Home', 'Away'])
ax[0].set_title('Average number of goals per match over team\'s venue')
ax[0].set_ylabel('Number of goals')

sns.barplot(x=['Home', 'Away', 'Draw'], y=[home_win, away_win, draw_count],
ax=ax[1], palette=['blue', 'red', 'grey'], hue = ['Home', 'Away', 'Draw'])
ax[1].set_title('Match outcome over team\'s venue')
ax[1].set_ylabel('Number of matches')
# Set y limit a bit higher
ax[1].set_ylim(0, home_win * 1.1)

# Add percentage of total match
for p in ax[1].patches:
    percentage = f'{100 * p.get_height() / total_matches:.1f}%'
    x = p.get_x() + p.get_width() / 2
    y = p.get_height()
    ax[1].text(x, y, percentage, ha='center', va='bottom')

plt.savefig('figures/venue_effect.png')
```



6. EDA of both datasets

6.a. Imports

```
import pandas as pd
from matplotlib import pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
import matplotlib.colors as mcolors
import matplotlib.cm as cm
import seaborn as sns

sns.set_theme(style = 'ticks', palette = 'pastel')
plt.rcParams['figure.autolayout'] = True
plt.rcParams['savefig.bbox'] = 'tight'
# Remove comment for saving figures
sns.set_context("paper")
# Define fig saving context
plt.rcParams['savefig.dpi'] = 300
plt.rcParams['savefig.bbox'] = 'tight'
```

```
plt.rcParams['savefig.directory'] = 'figures'

import numpy as np
from IPython.display import display, Markdown, HTML

from datetime import datetime

from utils.utils import filter_team, league_team, team_league, unique_teams,
unique_teams_coach_change
# league_team : league -> [team]
# team_league : team -> league
# unique_teams : [all teams]
# unique_teams_coach_change : [all teams that have had a coach change]

import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
```

6.b. Loading data

```
match_results = pd.read_csv('data/match_results.csv', parse_dates=['date'])
head_coach = pd.read_csv('data/head_coach.csv', parse_dates=['appointed',
'end_date'])

match_results.head()
```

	league	country	season_year	date	home_team	home_goals	away_team	away_goals
0	Premier League	England	2018	2017-08-11	Arsenal	4.0	Leicester City	3.0
1	Premier League	England	2018	2017-08-12	Watford	3.0	Liverpool	3.0
2	Premier League	England	2018	2017-08-12	Crystal Palace	0.0	Huddersfield	3.0
3	Premier League	England	2018	2017-08-12	West Brom	1.0	Bournemouth	0.0
4	Premier League	England	2018	2017-08-12	Chelsea	2.0	Burnley	3.0

```
for team in unique_teams:
    if filter_team(team).shape[0] !=
filter_team(team).drop_duplicates().shape[0]:
        print(f"Team {team} has more than one match in the same day")
# No team played more than one match in the same day (ie. (date, team) can be
index of match_result)
```

6.c. Basic plots

```
# Useful to add xtick months to dayofyear plot
months = pd.date_range('2022-01-01', '2022-12-31',
freq='M').strftime('%b').tolist()
days = np.linspace(1, 365, num=12, dtype=int)
```

6.d. Match related plots

```
def plot_team_result_ratio(league, team):
    """ Plot team's win ratio, draw ratio, lose ratio over time
    (win : pale green, draw : light grey, lose : pale red)"""
```

```
    team_result = filter_team(team)
    # Cumulative sum of each kind of result
    team_result['win'] = team_result['result'].apply(lambda x: 1 if x == 'win'
else 0)
    team_result['draw'] = team_result['result'].apply(lambda x: 1 if x == 'draw'
else 0)
    team_result['lose'] = team_result['result'].apply(lambda x: 1 if x == 'lose'
else 0)
    team_result['total'] = team_result['win'] + team_result['draw'] +
team_result['lose']

    # Ratio sum of wins, draws and loses
    team_result['win_ratio'] = team_result['win'].cumsum() /
team_result['total'].cumsum()
    team_result['draw_ratio'] = team_result['draw'].cumsum() /
team_result['total'].cumsum()
    team_result['lose_ratio'] = team_result['lose'].cumsum() /
team_result['total'].cumsum()

    fig, ax = plt.subplots(figsize=(10, 6))

    ax.plot(team_result.index, team_result['win_ratio'], color='yellowgreen',
linewidth=2)
    ax.plot(team_result.index, team_result['draw_ratio'], color='lightgrey',
linewidth=2)
    ax.plot(team_result.index, team_result['lose_ratio'], color='orangered',
linewidth=2)

    # Add discrete vertical line for mean of values
    ax.axhline(y=team_result['win_ratio'].mean(), color='yellowgreen',
linestyle='--', linewidth=1)
    ax.axhline(y=team_result['draw_ratio'].mean(), color='lightgrey',
linestyle='--', linewidth=1)
    ax.axhline(y=team_result['lose_ratio'].mean(), color='orangered',
linestyle='--', linewidth=1)

    # Head Coach change
    head_coach_team = head_coach[head_coach['team'] == team].copy()

    # Earliest Head Coach (some coach were stretching the plot a lot)
    min_label = team_result.index.min() - pd.Timedelta(days=150)
    min_head_coach = head_coach_team.appointed.min()
    if min_head_coach < min_label:
        earliest_head_coach_name = head_coach_team[head_coach_team['appointed']
== min_head_coach]['coach_name'].values[0]

        ax.axvline(x = min_label, color='black', linestyle='--', linewidth=1)
        ax.text(min_label + pd.Timedelta('10 days'), 0.5,
f'{earliest_head_coach_name} since {datetime.strftime(min_head_coach, "%d/%m/
%Y")}', rotation=90, verticalalignment='center')

        head_coach_team = head_coach_team[head_coach_team['appointed'] >
min_head_coach]

    for index, row in head_coach_team.iterrows():
        ax.axvline(x=row['appointed'], color='black', linestyle='--',
linewidth=1)
```

```
        ax.text(row['appointed'] + pd.Timedelta('10 days'), 0.5,
row['coach_name'], rotation=90, verticalalignment='center')

    ax.set_title(f"Ratios de résultats de {team} en {league} au fil du temps")
    ax.set_xlabel("Date")
    ax.set_ylabel("Ratio")
    ax.legend(['Victoires', 'Matchs nuls', 'Défaites'], loc='best')
    plt.show()

# plot_team_result_ratio('Ligue 1', 'Marseille');
```

```
import ipywidgets as widgets

league_widget = widgets.Dropdown(
    options = sorted(match_results['league'].unique().tolist()),
    description='Ligue:',
)

team_widget = widgets.Dropdown(
    options = sorted([team for team in unique_teams if team in
league_team[league_widget.value]]),
    description='Équipe:',
)

head_coach_change_widget = widgets.Checkbox(
    value=False,
    description="Changement d'entraîneur"
)

def update_team_options(*args):
    team_widget.options = league_team[league_widget.value]
    if head_coach_change_widget.value:
        team_widget.options = [team for team in team_widget.options if team in
unique_teams_coach_change]

league_widget.observe(update_team_options, 'value')
head_coach_change_widget.observe(update_team_options, 'value')

def plot_team(league, team, head_coach_change = None):
    plot_team_result_ratio(league, team)

widgets.interact(plot_team, league = league_widget, team = team_widget,
head_coach_change = head_coach_change_widget);
```

```
interactive(children=(Dropdown(description='Ligue:', options=('Bundesliga', 'La
Liga', 'Ligue 1', 'Premier Lea…
```

7. DZKAD

```
import pandas as pd

# Create a dataset that contains match results (win, draw, loss) and days since
head coach was appointed

# rows of match_result contains team1, team2, home_team, home_goals, away_team,
away_goals
match_results = pd.read_csv('data/match_results.csv', parse_dates=['date'])

## 1. Create home_results, away_results
```

```
def return_result(goal1, goal2):
    if goal1 > goal2:
        return 'win'
    elif goal1 < goal2:
        return 'loss'
    else:
        return 'draw'

match_results['home_result'] = match_results.apply(lambda x:
return_result(x['home_goals'], x['away_goals']), axis=1)
match_results['away_result'] = match_results.apply(lambda x:
return_result(x['away_goals'], x['home_goals']), axis=1)

# 2. Transform match into 2 separate rows relative to each team
home_results = match_results[['date', 'home_team', 'home_result']]
home_results.columns = ['date', 'team', 'result']

away_results = match_results[['date', 'away_team', 'away_result']]
away_results.columns = ['date', 'team', 'result']

match_results = pd.concat([home_results, away_results], axis=0)

# 3. Add a column that contains the days since the head coach was appointed for
that team

# head_coach contains team, coach_name, appointed, end_date
head_coach = pd.read_csv('data/head_coach.csv', parse_dates=['appointed',
'end_date'])
head_coach = head_coach[['team', 'league', 'appointed', 'coach_name',
'end_date']]

# Investigate non matching rows between match_results and head_coach
no_match = pd.merge(match_results, head_coach, on='team', how='outer')
match_without_coach =
no_match[no_match['appointed'].isna()].groupby('team').count()

print(f"Number of matches without a head coach: {match_without_coach.shape[0]}")
print("Team without head coach for some matches:")
print(", ".join(match_without_coach.index.unique()))

print("All coach have a matching team in teams result : ",
no_match[no_match['team'].isna()].shape[0] == 0)
```

```
Number of matches without a head coach: 69
Team without head coach for some matches:
Ajaccio, Amiens, Angers, Arminia, Aston Villa, Auxerre, Benevento, Bochum,
Bordeaux, Brentford, Brescia, Brest, Caen, Cardiff City, Chievo, Clermont Foot,
Crotone, Cádiz, Dijon, Düsseldorf, Eibar, Elche, Espanyol, Granada, Greuther
Fürth, Guingamp, Hamburger SV, Hannover 96, Heidenheim, Hellas Verona, Hertha
BSC, Holstein Kiel, Huddersfield, Huesca, Köln, La Coruña, Las Palmas, Lecce,
Leeds United, Leganés, Leicester City, Lens, Levante, Lorient, Mallorca, Metz,
Málaga, Norwich City, Nîmes, Nürnberg, Osasuna, Paderborn 07, Parma, SPAL,
Saint-Étienne, Salernitana, Sampdoria, Schalke 04, Sheffield Utd, Southampton,
Spezia, Stoke City, Swansea City, Troyes, Union Berlin, Valladolid, Venezia,
Watford, West Brom
All coach have a matching team in teams result :  True
```

```
# Merge the results and head_coach DataFrames on the 'team' column
merged = pd.merge(match_results, head_coach, on='team', how='inner')
```

```
# Filter the rows based on the 'date' and 'appointed' columns
filtered = merged[(merged['appointed'] <= merged['date']) &
                  ((merged['end_date'] > merged['date']) |
(merged['end_date'].isna()))]

check = filtered.groupby(['team', 'date']).size().reset_index(name='counts')
if check['counts'].max() >= 1:
    team_with_overlapping_coach = check[check['counts'] >= 2]['team'].unique()
    print(f"Some teams have multiple head coach at the same time:
{' ,'.join(team_with_overlapping_coach)}")
    display(head_coach[head_coach['team'].isin(team_with_overlapping_coach)])

    # Drop teams with overlapping head_coach
    filtered = filtered[~filtered['team'].isin(team_with_overlapping_coach)]
```

```
Some teams have multiple head coach at the same time: Reims
```

	team	league	appointed	coach_name	end_date
294	Reims	Ligue 1	2022-10-13	Will Still	NaT
295	Reims	Ligue 1	2021-06-23	Óscar García	2022-10-13
296	Reims	Ligue 1	2018-07-01	Sébastien Desmazeau	2019-03-30
297	Reims	Ligue 1	2017-05-22	David Guion	2021-05-25

```
# Calculate the number of days since the head coach was appointed
filtered['days_in_post'] = (filtered['date'] - filtered['appointed']).dt.days

print(f"{filtered.shape[0]} matches out of {match_results.shape[0]} remains
after excluding matches where we don't have information on head coach or there
is overlapping head coaches.")

filtered.sort_values(['team', 'date']).head()
```

```
11637 matches out of 18296 remains after excluding matches where we don't have
information on head coach or there is overlapping head coaches.
```

	date	team	result	league	appointed	coach_nam	end_date	days_in_post
29707	2017-12-04	Alavés	win	LaLiga	2017-12-01	Abelardo	2019-06-30	3
4183	2017-12-08	Alavés	win	LaLiga	2017-12-01	Abelardo	2019-06-30	7
29753	2017-12-16	Alavés	loss	LaLiga	2017-12-01	Abelardo	2019-06-30	15
4252	2017-12-21	Alavés	win	LaLiga	2017-12-01	Abelardo	2019-06-30	20
29827	2018-01-07	Alavés	loss	LaLiga	2017-12-01	Abelardo	2019-06-30	37

```
# Exclude matches with days_in_post > 2000
filtered = filtered[filtered['days_in_post'] <= 2000]
# Save as match_results2.csv
match_results = filtered[['date', 'league', 'team', 'result', 'days_in_post']]
match_results.to_csv('data/match_results2.csv', index=False)
```

8. Stat analysis

8.a.i. *Imports:*

```
import pandas as pd
import numpy as np
```

```
from matplotlib import pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
import matplotlib.colors as mcolors
import matplotlib.cm as cm
import seaborn as sns
from IPython.display import display, Markdown, HTML
from datetime import datetime

sns.set_theme(style = 'ticks', palette = 'pastel')
plt.rcParams['figure.autolayout'] = True
plt.rcParams['savefig.bbox'] = 'tight'
sns.set_context("paper")
# Define fig saving context
plt.rcParams['savefig.dpi'] = 300
plt.rcParams['savefig.bbox'] = 'tight'
plt.rcParams['savefig.directory'] = 'figures'
```

8.a.ii. *Loading data:*

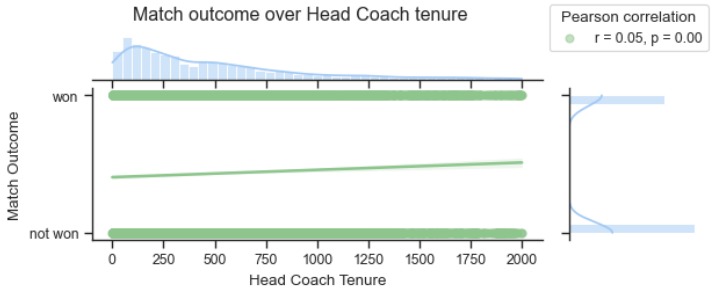
```
match_results = pd.read_csv('data/match_results2.csv', parse_dates=['date'])
match_results.shape
```

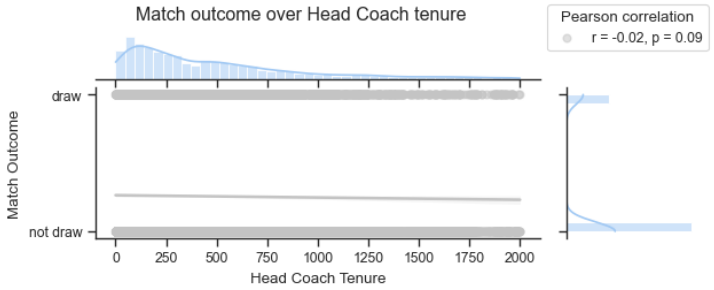
```
(10921, 5)
```

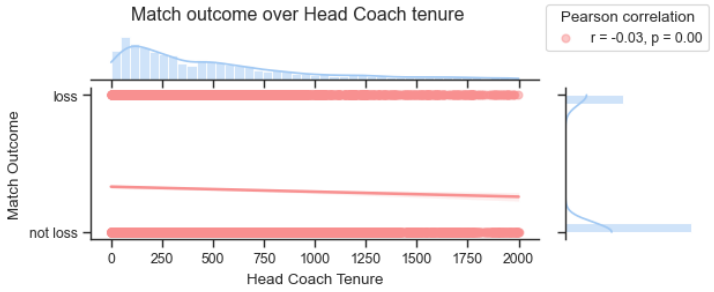
```
match_results['win'] = match_results['result'].apply(lambda x: 1 if x == 'win'
else 0)
match_results['loss'] = match_results['result'].apply(lambda x: 1 if x == 'loss'
else 0)
match_results['draw'] = match_results['result'].apply(lambda x: 1 if x == 'draw'
else 0)
```

```
from sklearn.linear_model import LinearRegression
from scipy.stats import pearsonr

def plot_match_outcome_over_coach_tenure(data, y_value, y_label, color):
    # Create a jointplot
    # g = sns.jointplot(data=data, x='days_in_post', y=y_value, kind='reg',
ratio = 3, marginal_ticks = False)
    g = sns.jointplot(data=data, x='days_in_post', y=y_value, kind='reg',
                      scatter_kws={'alpha':0.5, 'color':
sns.light_palette(color, as_cmap=True)(0.4)},
                      line_kws={'color': sns.light_palette(color, as_cmap=True)
(0.4)},
                      ratio = 3, marginal_ticks = False)
    g.figure.set_figwidth(6)
    g.figure.set_figheight(2)
    g.figure.suptitle(f'Match outcome over Head Coach tenure', x = 0.4, y = 1.1)
    g.set_axis_labels('Head Coach Tenure', 'Match Outcome')

    # Legend
    r, p = pearsonr(data['days_in_post'], data[y_value])
    legend = g.ax_joint.legend([f'r = {r:.2f}, p = {p:.2f}'], loc='upper left',
bbox_to_anchor=(1, 1.6))
    legend.set_title("Pearson correlation")

    # Set y-axis tick
    g.ax_joint.set_yticks([0, 1])
    g.ax_joint.set_yticklabels(['not ' + y_label, y_label])
```

```
    # Save the figure
    g.savefig(f'figures/{y_value}_over_coach_tenure.png')

plot_match_outcome_over_coach_tenure(match_results, 'win', 'won', 'green')
plot_match_outcome_over_coach_tenure(match_results, 'draw', 'draw', 'gray')
plot_match_outcome_over_coach_tenure(match_results, 'loss', 'loss', 'red')
```





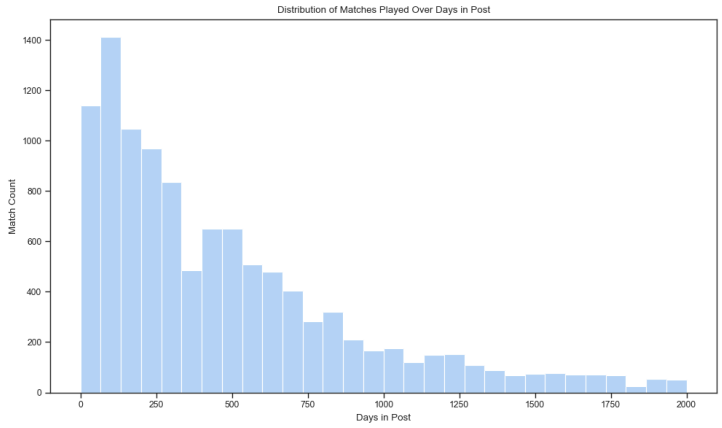



```
match_results_bis = match_results.groupby('days_in_post').agg({'win': 'mean',
'draw': 'mean', 'loss': 'mean', 'result': 'count'})
match_results_bis.columns = ['win_rate', 'draw_rate', 'loss_rate',
'match_count']
# Add missing days between the first and last day
match_results_bis =
match_results_bis.reindex(range(match_results_bis.index.min(),
match_results_bis.index.max() + 1), fill_value=0)

def weighted_rolling_mean(data, weights, window_size=30):
    def weighted_mean(x):
        return np.average(data.loc[x.index], weights=weights.loc[x.index])

    return data.rolling(window_size, min_periods=1).apply(weighted_mean,
raw=False)

match_results_bis['win_rate_smooth'] =
```

```
weighted_rolling_mean(match_results_bis['win_rate'],
match_results_bis['match_count'])
match_results_bis['draw_rate_smooth'] =
weighted_rolling_mean(match_results_bis['draw_rate'],
match_results_bis['match_count'])
match_results_bis['loss_rate_smooth'] =
weighted_rolling_mean(match_results_bis['loss_rate'],
match_results_bis['match_count'])

# Ensures it sums to 1
(match_results_bis['win_rate_smooth'] + match_results_bis['draw_rate_smooth'] +
match_results_bis['loss_rate_smooth']).value_counts()
```

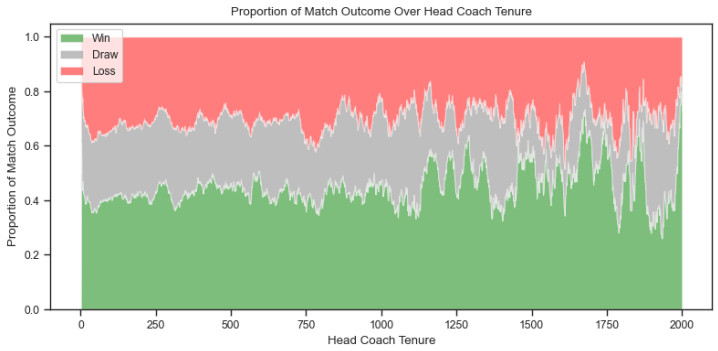
```
1.0    1902
1.0      99
Name: count, dtype: int64
```

```
# Create a histogram of 'match_count' over 'days_in_post'
plt.figure(figsize=(10, 6))
sns.histplot(data=match_results_bis, x='days_in_post', weights='match_count',
bins=30)
plt.xlabel('Days in Post')
plt.ylabel('Match Count')
plt.title('Distribution of Matches Played Over Head Coach Tenure')
plt.show()
```




```
def plot_proportion_over_coach_tenure(data):
    # Create a figure
    plt.figure(figsize=(8, 4))

    plt.stackplot(data.index, data['win_rate_smooth'], data['draw_rate_smooth'],
data['loss_rate_smooth'], labels=['Win', 'Draw', 'Loss'], colors=['green',
'gray', 'red'], alpha=0.5)

    # Add legend and labels
    plt.legend(loc='upper left')
    plt.xlabel('Head Coach Tenure')
    plt.ylabel('Proportion of Match Outcome')
    plt.title('Proportion of Match Outcome Over Head Coach Tenure (30-day
weighted rolling average)')

    # Save the figure
    plt.savefig('figures/proportion_over_coach_tenure.png')
    plt.show()
```

```
plot_proportion_over_coach_tenure(match_results_bis)
```



8.b. Correlation between days in post and teams performance

- could indicate that club keeps their well performing head-coaches
- could indicate that head coaches performance improve after time either because:
  - early low performance : coaches need some time once they are appointed to reach previous team performance
  - long term improvement of performance

9. Conclusion

9.a. Conclusion

End of paper

References

Rocaboy, Y., & Pavlik, M. (2020). Performance Expectations of Professional Sport Teams and In-Season Head Coach Dismissals—Evidence from the English and French Men's Football First Divisions. *Economies*, *8*(4), 82–83. https://doi.org/10.3390/economies8040082