Data Visualization Report

1. Introduction

For this data visualization assignment, European Soccer Database (ESD) has been chosen to draw out some of the facts about the matches played by each team and some of the facts about the team. Several matches were played from 2009 to 2016. So firstly, this assignment addresses combining of geojson data of Europe continent with ESD data to figure the matches played in each country.

Furthermore, we calculate the total number of matches played, total number matches won, lost, or drawn during the period from 2008 to 2016 by each team and visualize them in stacked bar graph . This assignment also explores about team strategies used by each team during a year. By analyzing the visual graphs, we can explore what strategies work best for getting best outcomes for a team. Alternatively, to put it into other words, which strategy could be used for a team to win more games.

2. Dataset

2.1 European Soccer Database

The dataset is taken from Kaggle website. The data was already in the SQLite database. (This data is extracted from different sources through data mining techniques. The project can be found here) .So a little or no data cleaning was required. However, data cleaning was required to clean data present in rows. For instance, some of the team names contained numbers. So regular expressions were used to cleanse and update back the row values. Additionally, the data is present different tables. So relevant SQL queries are written joining tables on relevant column ids to extract the information. The table below shows the table name and the column names present in different tables.

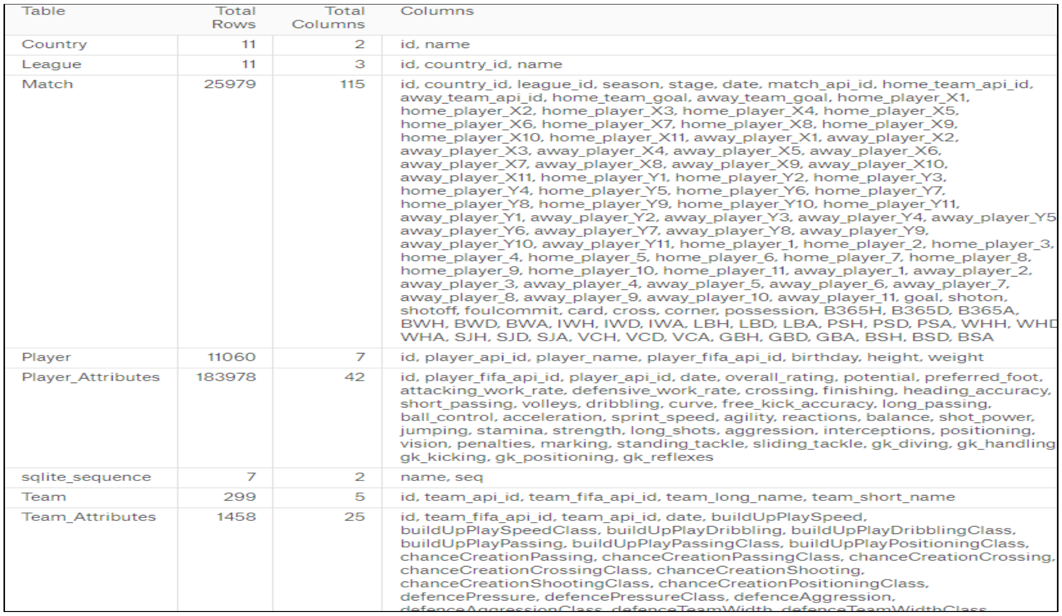


Table	Total Rows	Total Columns	Columns
Country	11	2	id, name
League	11	3	id, country_id, name
Match	25979	115	id, country_id, league_id, season, stage, date, match_api_id, home_team_api_id, away_team_api_id, home_team_goal, away_team_goal, home_player_X1, home_player_X2, home_player_X3, home_player_X4, home_player_X5, home_player_X6, home_player_X7, home_player_X8, home_player_X9, home_player_X10, home_player_X11, away_player_X1, away_player_X2, away_player_X3, away_player_X4, away_player_X5, away_player_X6, away_player_X7, away_player_X8, away_player_X9, away_player_X10, away_player_X11, home_player_Y1, home_player_Y2, home_player_Y3, home_player_Y4, home_player_Y5, home_player_Y6, home_player_Y7, home_player_Y8, home_player_Y9, home_player_Y10, home_player_Y11, away_player_Y1, away_player_Y2, away_player_Y3, away_player_Y4, away_player_Y5 away_player_Y6, away_player_Y7, away_player_Y8, away_player_Y9, away_player_Y10, away_player_Y11, home_player_1, home_player_2, home_player_3, home_player_4, home_player_5, home_player_6, home_player_7, home_player_8, home_player_9, home_player_10, home_player_11, away_player_1, away_player_2, away_player_3, away_player_4, away_player_5, away_player_6, away_player_7, away_player_8, away_player_9, away_player_10, away_player_11, goal, shoton, shotoff, foulcommit, card, cross, corner, possession, B365H, B365D, B365A, BWH, BWD, BWA, IWH, IWD, IWA, LBH, LBD, LBA, PSH, PSD, PSA, WHH, WHD WHA, SJH, SJD, SJA, VCH, VCD, VCA, GBH, GBD, GBA, BSH, BSD, BSA
Player	11060	7	id, player_api_id, player_name, player_fifa_api_id, birthday, height, weight
Player_Attributes	183978	42	id, player_fifa_api_id, player_api_id, date, overall_rating, potential, preferred_foot, attacking_work_rate, defensive_work_rate, crossing, finishing, heading_accuracy, short_passing, volleys, dribbling, curve, free_kick_accuracy, long_passing, ball_control, acceleration, sprint_speed, agility, reactions, balance, shot_power, jumping, stamina, strength, long_shots, aggression, interceptions, positioning, vision, penalties, marking, standing_tackle, sliding_tackle, gk_diving, gk_handling gk_kicking, gk_positioning, gk_reflexes
sqlite_sequence	7	2	name, seq
Team	299	5	id, team_api_id, team_fifa_api_id, team_long_name, team_short_name
Team_Attributes	1458	25	id, team_fifa_api_id, team_api_id, date, buildUpPlaySpeed, buildUpPlaySpeedClass, buildUpPlayDribbling, buildUpPlayDribblingClass, buildUpPlayPassing, buildUpPlayPassingClass, buildUpPlayPositioningClass, chanceCreationPassing, chanceCreationPassingClass, chanceCreationCrossing, chanceCreationCrossingClass, chanceCreationShooting, chanceCreationShootingClass, chanceCreationPositioningClass, defencePressure, defencePressureClass, defenceAggression, defenceAggressionClass, defenceTeamWidth, defenceTeamWidthClass,

Figure 1

For this assignment, the tables chosen are Match, Team, Team_Attributes, League and Country.

2.2. Europe Continent GeoJSON

The GeoJSON data is used to show number of matches played in different countries.This data is collected from this website. From the above figure, we can conclude that country table can be combined with GeoJSON data to show the stats. GeoJSON data is combined with country table by country names. However, care must be taken to combine GeoJSON data with country data as the country name should match. So, spellings have to match and if there any trailing spaces, must also be removed.

3. Analysis and Visualization

Before dwelling into the process of creation of graphs, let's discuss about our data and some of the thought process involved in choosing graphs.

At first glance of Match table, we notice that it has time series and so one would consider that a line chart would be the best option for showing some interesting facts. So one might choose to show 25000 matches played over the period. Yes, I would agree that we would end up with a pretty graph. However, it would not be telling us any story.

So after some analysis, a conclusion was made to show stacked bar graphs of each team's win, lose or draw percentages over the period.

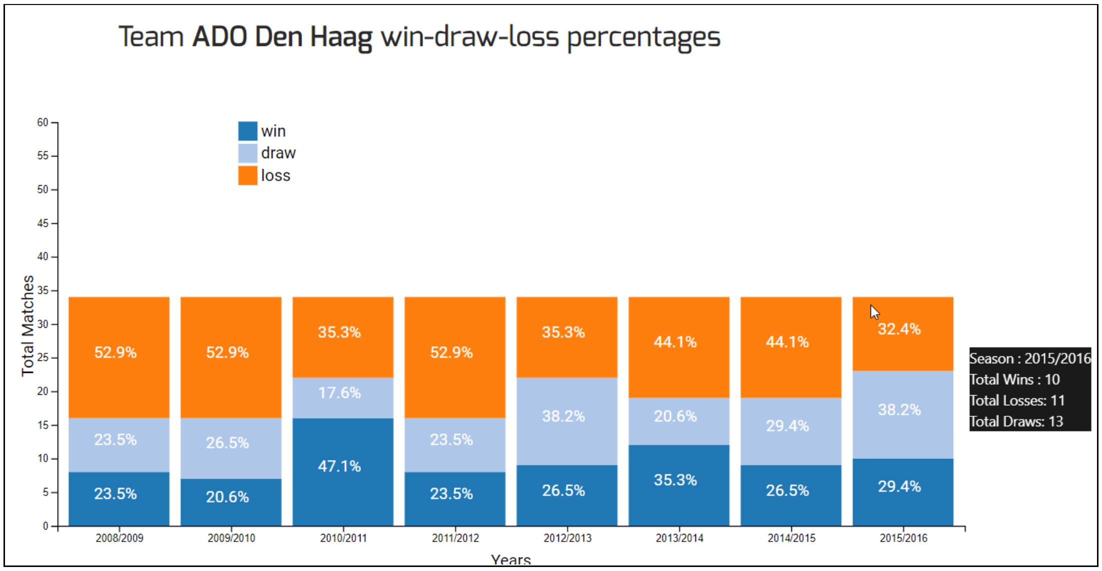


Figure 2

As we can see above, this data could be further used to plan strategies for winning games in the upcoming seasons.

Further, it would be great to show what kind of strategies were employed by a certain team in a given year. By doing so we would come to know which strategies worked and which did not work. So a spider chart is used to show different strategies of a team.

4. Technical Details

4.1. Database

As the data is in SQLite database, to view that data, DB Browser for SQLite is used. This tool is very simple to use and it allows us to view table structure, table values and write SQL queries. More information about the tool could be found here.

4.2 Web Application

A simple web application is built to retrieve the data from database as JSON structured data and to transform the data into visual representation on the frontend side. On the frontend side, Angular 4 and D3.js is used to show the visual graphs.

For transforming the data present in tables to JSON and to retrieve the data to the frontend side, Flask framework is used. Flask is a very simple framework built in python language which is best suited for REST API.

The server side code is built using PyCharm community version IDE, which is best suited for building python related applications. In addition, to write frontend code, Visual Code from Microsoft was used. Visual Code is best suited for writing any JavaScript, HTML, CSS related code these days.

5. Process.

To show a team's total win, loss and draw the tables team, matches were combined. Below is the query, which was used to get the result.

```
select t.team_long_name as team_name,
count(m.id) as total_matches,
m.season,m.home_team_api_id as home,m.away_team_api_id as away,
SUM(m.home_team_goal) as total_home_team_goals,
SUM(m.away_team_goal) as total_away_team_goals,
SUM(CASE
    WHEN (m.home_team_api_id = team_api_id and m.home_team_goal > m.away_team_goal and
m.home_team_goal <> m.away_team_goal) OR
         (m.away_team_api_id = team_api_id and m.away_team_goal > m.home_team_goal and
m.home_team_goal <> m.away_team_goal)
    THEN 1 ELSE 0 END) as win,
SUM(CASE
    WHEN (m.home_team_api_id = team_api_id and m.home_team_goal < m.away_team_goal and
m.home_team_goal <> m.away_team_goal) OR
         (m.away_team_api_id = team_api_id and m.away_team_goal < m.home_team_goal and
m.home_team_goal <> m.away_team_goal)
    THEN 1 ELSE 0 END) as loss,
SUM(CASE WHEN m.home_team_goal = m.away_team_goal THEN 1 ELSE 0 END) as draw
```

```
from match m inner join team t
on t.team_api_id = m.home_team_api_id OR t.team_api_id = m.away_team_api_id
group by m.season,t.team_api_id
having t.team_api_id =
```

The results from above query was then called as an API in python framework. Later this API is called through an AJAX function to get results dynamically and show the results in a stacked bar graph

Upon clicking on stacked bar graph, AJAX calls are made to the server to get team attributes. Then these team attributes are shown in spider graph to show different strategies employed during a particular year(shown in figure 2). To get the team attributes team table was combined with team attributes table to get the result.

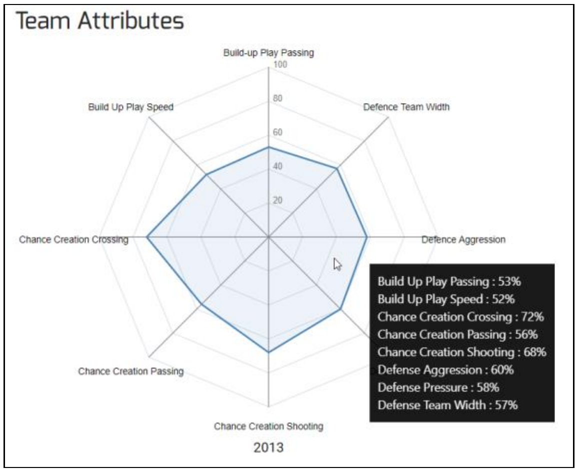


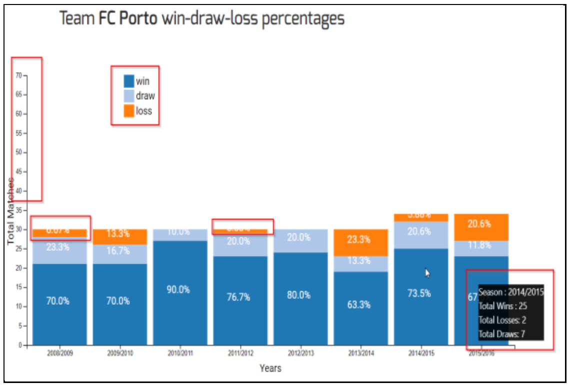
Figure 3

To get the total matches played in different countries, country table, matches table and geojson data was combined to get the result.

6. Result Analysis

6.1 Stacked Bar Chart

As shown in figure 2, we have achieved to show win/loss/draw percentages over a period. The Y-axis shows the total number of matches played and the X-axis shows the Year. One can notice that X-axis is shown xxxx/xxxx year format. This is because leagues usually start from June.



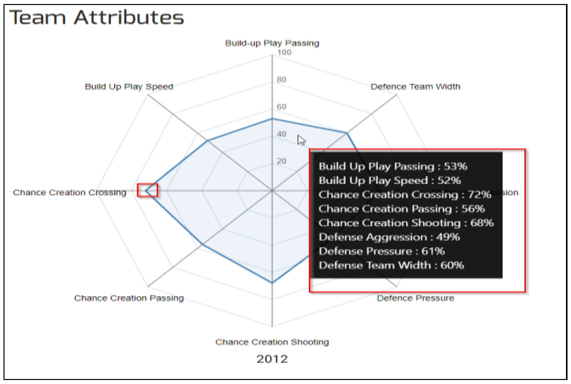


Figure 4

Figure 5

6.1.1 Challenges and Difficulties

As we can see from the above figure 4, I couldn't scale down the Y-Axis from showing too many ticks. The legend should have been on the right corner but that couldn't be achieved as well. Also some of the loss percentage text were trimmed off as the height of the bar was not sufficient. Also, observe that black popup which is appearing very far away from mouse, this has been deliberately done because it was difficult to navigate to other stacked bars.

6.2 Spider chart

As shown in figure 3, we are able to map the team attributes onto a spider chart. The texts on each corner of the chart shows the team attribute. Each attribute is mapped to a vertex of the spider chart.

6.1.2 Challenges and Difficulties

Drawing a spider chart is very complicated in d3js. To be very honest, most of the code that has been written is open sourced. Only few lines of the code were changed to show the spider chart. As shown in figure 5, the inner vertices should have been marked with the percentage. However, I was unable to figure out how to draw on the chart. Second thing is that popup is blocking the other vertices of the chart.

6.3 Map

Collected GeoJSON data is combined with Country table to show the matches played in different countries. The density of the color changes according the matches played in each country.

7. Principles of Data Visualization

7.1 Goals

1. Information – The title is given to bar chart like for example "Team FC Porto win-draw-loss percentages" focuses on giving the info about win/loss/draw streak of a team.
2. Persuasion – The popup upon hovering of mouse on stacked bars makes the user to look onto the info present in popup.
3. Entertainment - The stacked bar chart and animated once a user chooses a team from a dropdown list.

7.2 Kirk’s Principles

1. Trustworthy – Stacked bar, Spider chart and Map are not using any unnecessary colors. The colors that have been used in stacked bar are from d3 js category20
2. Accessible – Stacked bar graphs are easily accessible to users by simply clicking on side panel on clicking on dropdown list of teams.
3. Elegant – The graphs mostly emphasize on style rather thorough.

7.3 Gestalt Laws

1. Connectivity – Once the user clicks on stacked bars, immediately spider chart is shown below the stacked bar chart
2. Proximity – The stacked bars the closely placed showing that they all are related to one another

8. Project Details

1. The demo application is here - https://valekar.herokuapp.com/#/pages .
2. Code for the project can be found on github.com. Here is the link - https://github.com/valekar/data_visualization
3. The video can be found here - https://youtu.be/TNwltwYQtIM