

# Interactive Football Match Visualization

Bram Silue, Kevin Sam

bram.silue@vub.be — 0538225

kevin.sam@vub.be — 0558882

*Information Visualization*  
*Vrije Universiteit Brussel*

May 28, 2023

## Abstract

This paper has been made in the context of a course on Information Visualization. This course is a Master's level course given at the Vrije Universiteit Brussel (VUB) in the *Master of Applied Sciences and Engineering in Computer Science*.

## 1 Introduction

As part of the course *Information Visualization*, students are expected to work on a project in groups of three. The goal of this project is to develop a visualization of a particular dataset, following the theoretical principles learned throughout the course. First, the students are expected to choose a dataset and apply pre-processing techniques if needed. Next, a visualization must be made with a clear target audience in mind. Finally, the students must present an evaluation for their visualization.

Our team has chosen to create a visualization app in the form of a web application that aims to help football coaches and scouts with the analysis of football players and football matches. Indeed, the *task* of our app is to provide them with a way to gain insights into the play styles of certain players, and patterns in the match, allowing them to determine how to coach specific players or whether or not to recruit a particular player. As a bonus, our app also functions as a fun tool to play with for the average football enthusiast.

## 2 Dataset

We shall now look at *what* data our app must visualize. Our chosen dataset was obtained from Kaggle<sup>1</sup>. The name of the dataset is *Soccer match event dataset* (Pappalardo et al., 2019). This dataset contains data about the World Cup of 2018, the European Cup of 2016, and five national soccer competitions played in the year 2017 and 2018 in Europe: The Spanish first division, the Italian first division, the English first division, the German first division, and the French first division. In our project, our focus is on the World Cup of 2018. nah but The dataset is made out of different files, each containing different kinds of information about the football matches. In total, it contains 27 different files. Some of the files contain similar information. For example, the events are categorized based on the football competition they occurred in.

This dataset contains an overwhelming amount of information, and we will not use every file in our visualization process. We decided to remove some of the unnecessary files from our dataset as a first pre-processing step. We will not present these files in more detail as the information they contain is irrelevant to our visualization.

The files that we did keep contain information about the events, the players, the matches, and the player rank scores.

---

<sup>1</sup><https://www.kaggle.com/>

## 2.1 Event Files

The Event files contain information about the matches in the form of *events*. These files take the form of tables that contain all the events that occurred during the competition, where an event refers to some sort of ball touch. Each event (i.e. row) in the file is characterized by many attributes (i.e. columns), many of which were important for our visualization. The attributes that we used are:

- **id**: uniquely identifies the specific event.
- **eventName**: could be a pass, foul, shot, duel, free kick, offside, or touch.
- **subEventName**: each event is an umbrella for subevents that provide more detail. For instance, the event “Pass” has subevents “Simple Pass”, “High Pass”, “Head Pass”, etc.
- **eventSec**: denotes the time of occurrence for an event measured in seconds since the start of the current half.
- **matchId**: refers to the identifier of the match associated with the event, matching the “wyId” attribute in the match dataset.
- **matchPeriod**: indicates the period of the match, such as “1H” for the first half, “2H” for the second half, “E1” for the first extra time, “E2” for the second extra time, or “P” for penalties.
- **playerId**: corresponds to the identifier of the player responsible for acting out the event, matching the “wyId” attribute in the player dataset.
- **teamId**: corresponds to the identifier of the player’s team, matching the “wyId” attribute in the team dataset.
- **pos\_orig\_x**: represents the x-coordinate of the origin position.
- **pos\_orig\_y** : represents the y-coordinate of the origin position.
- **pos\_dest\_x** : represents the x-coordinate of the destination position.
- **pos\_dest\_y**: represents the y-coordinate of the destination position.

## 2.2 Match Files

The Match files contain information about the matches played in the different competitions. The file contains the following information:

- **wyId**: the identifier of the competition to which the match belongs.
- **dateutc**: specifies the date and time when the match starts in the format YYYY-MM-DD hh:mm:ss.
- **label**: contains the name of the two teams and the result of the match (e.g., “Belgium - France, 1 - 0”).
- **teamsData**: contains information about each team playing the match, including lineup, bench composition, substitutions, coach, and scores.

## 2.3 Players file

The Players file contains information on the different players in the different competitions. Among many attributes, those relevant to our application are:

- **wyId**: the identifier of the player.
- **shortName**: the short name of the player (e.g. “K. Mbappé”).
- **role**: the main role of the player, including the role’s name and two abbreviations.

- **currentTeamId**: the identifier of the team where the player plays. The identifier refers to the "wyId" attribute in the team file.
- **foot**: the preferred foot of the player.
- **height**: the height of the player in centimeters.
- **weight**: the weight of the player in kilograms.
- **birthDate**: the birth date of the player, in the format "YYYY-MM-DD."
- **passportArea**: the geographic area associated with the player's current passport.

## 2.4 Teams file

This file contains information about the different teams in the competition. These can be national teams or clubs. Within this file, we use the following information:

- **wyId**: the identifier of the team.
- **officialName**: the official name of the team, such as "Juventus FC."

## 2.5 PlayeRank file

This file contains information about how players performed in a certain match. This is measured using the PlayeRank score. Within this file, we use the following information:

- **goalScored**: the number of goals scored by the player in the match.
- **playerankScore**: the PlayeRank score of the player in the match.
- **matchId**: the identifier of the match.
- **playerId**: the identifier of the player.
- **minutesPlayed**: the number of minutes played by the player in the match.

## 2.6 Pre-processing

The pre-processing of the dataset was achieved using the Pandas framework in Python. In fact, all data manipulation across the app was performed with this framework.

### 2.6.1 Filtering files and attributes.

As previously mentioned, we do not use all the files present in the original dataset since they are not all relevant to our visualization. We only use the files discussed in this section, i.e.: the Event files, the Match files, the Players file the Teams file, and the PlayeRank file.

Furthermore, some of the files we want to use contain information that is not relevant to our use. Therefore, we decided to remove the unnecessary attributes to speed up any computation that may be required during the time that the application is run.

### 2.6.2 Correcting Event Positions

In the original dataset, some event positions are incorrect, causing the visualization to sometimes yield abnormal results. For example, there was a match that showed that almost all the passes in the match, went right through the middle of the attribute, creating a strange star pattern in the visualization. In theory, it is possible, but we took the time to analyze the match by watching its replay and we did not see this particular strategy being played during the match.

The solution we came up with is pretty intuitive. Every event that has an ending position that is not the same as the starting position of the next event, will be changed so that it effectively becomes the same. With this pre-processing step, the visualization of the events on the map is mostly correct. The solution may not always be optimal, however, because in some cases it might make an incorrect visualization. For example when we think of the case where the ball goes out

and the next event is a corner shot. At this moment, the ending position of the previous event is in most cases not the same as the starting position of the next event.

### 2.6.3 Actions

In the original dataset, there is no way to detect which action an event was part of. We define an action as a series of events, where one team keeps control of the ball. To annotate the different events with an action number, we first sorted the different events by the `eventSec` attribute. This attribute gives a time in seconds for when an event happened. After this step, the events are therefore chronologically sorted. To group the events together based on the action they occurred in, we check whether, in two consecutive events, there was a change in `teamId`. If there was, it means that the ball possession has changed and therefore that a new action has started. We do that for every event and we annotate the events with an action number accordingly.

## 3 The App

Based on the dataset we have discussed previously, it is easy to see that the data is overwhelmingly large and impossible to grasp in raw form from a human perspective due to the many hundreds of positional events per game. This is where our visualization app comes in. It provides an intuitive way to get a high-level understanding of the data and extract meaningful information.

### 3.1 Running the app

In order to run our visualization app, several Python packages must be installed. Doing so can be achieved by opening a terminal in the root directory of the project and running the command `'pip install -r requirements.txt'`, assuming Python 3.4 or later is already installed on your machine. Next, the app can be run by executing the command `'streamlit run code/main.py'` in that same terminal.

### 3.2 Technical Setup

The entire app was built in the programming language Python. The visualization tools that are at the core of our project are the Python frameworks *Streamlit* and *Plotly*. Streamlit was used to create the blueprint and foundation of our web application. It allowed for the creation of the main page which contains all the visualizations, and the sidebar which contains most of the settings that the user can specify with respect to those visualizations. The visualizations themselves are Plotly plots which were integrated inside the Streamlit page. As previously mentioned, all data manipulation is achieved with Pandas. Figure 1 below shows an overview of the entire app.

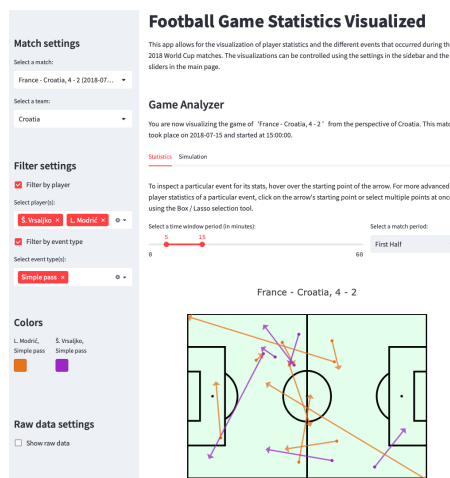


Figure 1: An overview of the app.

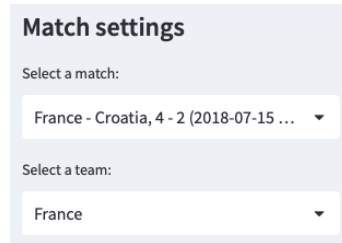
### 3.3 Features

Our visualization app is structured in two main areas. The first area is *Settings Selection* menu in the sidebar. This menu gives the user the ability to select the data to be visualized. The area is the main area called the **Game Analyzer**. The Game Analyzer menu contains two sub-menus, a **Statistics** sub-menu to observe the statistics of events and players, and a **Simulator** sub-menu to simulate the game.

#### 3.3.1 Settings Selection.

The bulk of the settings selection happens in the sidebar. The settings selection is divided into three main categories, which are the **Match settings**, the **Filter settings**, and the **Color settings**.

To use our app, a user will generally start by exploring the **Match settings** first. Under this section, the user shall select a particular match using the match selection drop-down box. After that, the user should select a team within that match that they would like to analyze. Figure 2 provides a look into this menu.

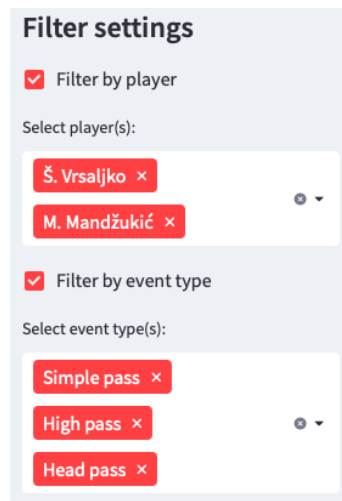


The screenshot shows a sidebar menu titled "Match settings". It contains two sections: "Select a match:" with a dropdown menu showing "France - Croatia, 4 - 2 (2018-07-15 ...)" and "Select a team:" with a dropdown menu showing "France".

Figure 2: The match settings menu in the sidebar of the app.

After setting those match settings, the user may move on to explore the **Filter settings** section. The user may use the **Filter by player** checkbox to decide whether or not they would like to visualize individual players. If this filter is disabled, then the entire team's events will be visualized. However, if this filter is enabled, then a player selection drop-down box will appear, which will allow the selection of multiple players to analyze.

After these settings have been decided upon, the user may use the **Filter by event type** checkbox to decide whether or not to filter the visualizations based on specific event types. Leaving this filter disabled will display all the different event types that occurred. When this filter is enabled, the user can select event types to visualize by selecting them in the event type drop-down box that appears. Figure 3 provides a look into the use of this menu.



The screenshot shows a sidebar menu titled "Filter settings". It contains two sections: "Filter by player" which is checked, with a dropdown menu showing "Š. Vrsaljko" and "M. Mandžukić"; and "Filter by event type" which is checked, with a dropdown menu showing "Simple pass", "High pass", and "Head pass".

Figure 3: The filter settings menu. In this example, two players and two event types have been selected.

Once events have been selected for visualization, the user will notice that the football pitch plot under the **Game Analyzer** section on the main page now provides a visualization of the different events in the game. Furthermore, the sidebar will now have expanded with a new settings selection section called **Color settings**. This section provides the possibility to customize the colors associated with each event-player pair, by clicking on the color box under the event-player pair name and selecting the desired color for that pair. Figure 4 showcases this menu.

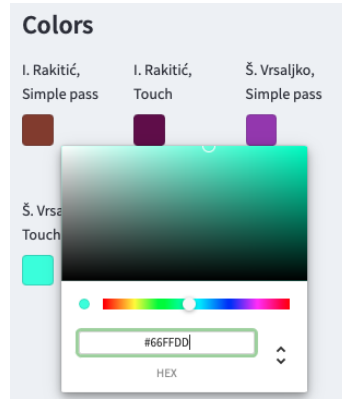


Figure 4: The color settings menu. In this example, the bottom-left color box was clicked, which opens up a color-picking tool.

Finally, the user may, if they want, enable a display of the raw data under the **Raw Data Settings** menu at the bottom of the sidebar, by checking the **Show raw data** checkbox. This will display the raw Pandas dataframe data used by the main visualizations in the main page. The dataframe will be made visible at the complete bottom of the main page. We included this as an option for more advanced users who want all the details of the data in raw form, though this is not necessarily something the average user would look at.

As previously mentioned, this web application was developed using Streamlit. As such, all elements of the sidebar were created using Streamlit components such as `selectbox`, `checkbox`, `multiselect`, etc.

### 3.3.2 Game Analyzer: Statistics.

Once the settings have been chosen, the first feature the user can interact with is the **Statistics** sub-menu under the Game Analyzer. In this section of the app, the user is greeted with a top-view visual representation of a football pitch. Above it, the user will find a two-way slider that allows for choosing the time frame within which the football events shall be visualized. Next to the slider, the user can choose which phase of the match should be visualized: The first or second half, the first or second extension, or the penalties. The events are shown as arrows on the pitch, each respecting the colors that were specified in the settings menu. Figure 5 showcases the **Statistics** sub-menu.

As shown on the right-hand side in Figure 5, hovering over the starting point of an event arrow will reveal details about the event and the player who performed it. More specifically, it reveals the role, age, height, weight, preferred foot, and country of origin of the player. It also reveals the type and position of the event. We gave the arrows a starting point to give a clear visual mark of where exactly an event started. The opacity of the point is higher than the rest of the arrow, which helps convey the intuition that those starting points are important, which is the case because they are the interactive component of the plot.

The implementation of this football pitch visualization was achieved using Plotly. The pitch itself was rendered using a package called `plotly-football-pitch`, and the arrows were then rendered on top of this attribute using Plotly annotations.

### Game Analyzer

You are now visualizing the game of 'France - Croatia, 4 - 2' from the perspective of Croatia. This match took place on 2018-07-15 and started at 15:00:00.

**Statistics** Simulation

To inspect a particular event for its stats, hover over the starting point of the arrow. For more advanced player statistics of a particular event, click on the arrow's starting point or select multiple points at once using the Box / Lasso selection tool.

Select a time window period (in minutes):



Select a match period:

First Half

France - Croatia, 4 - 2

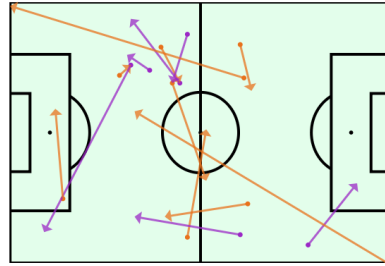
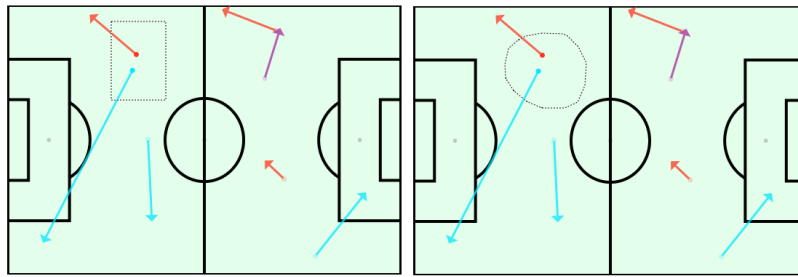


Figure 5: The **Statistics** sub-menu. In this example, the time slider is set to show events between minutes 10 and 17 for multiple players and actions.



(a) Box Select.

(b) Lasso Select.

Figure 6: Selecting events in the **Game Analyzer** under the **Statistics** sub-menu using the "Box Select" option on the left, and the "Lasso Select" option on the right.

### 3.3.3 Player Statistics Visualizer.

Next to the information about the events displayed on the attribute, the user can also get more information on the player that did the event, as can be seen in Figure 7. There are three different ways how the user can get to see more information about the player(s) that did the event(s). The first, and most straightforward way to do it is by clicking on the starting point of the event directly. This will immediately open the player statistics for the player that performed the event that was just clicked on.

The two other ways to do that are by using the "Box Select" and the "Lasso Select" options available in the Game Analyzer plot under the **Statistics** sub-menu. The selection using the "Box Select" option can be seen in Figure 6a and the selection using the "Lasso Select" option can be seen in Figure 6b.

The player statistics give more information about how the player performed during the match. It gives information about the number of goals that the player has scored during the match, the number of assists that the player did during the match, the number of red and yellow cards that the player has gotten during the match, and it also gives information about the playtime of the player and whether or not the player was in the original lineup. If the player was not in the original lineup, then it will show which player has been replaced instead. In addition to that, the user is

presented with a distribution plot that shows a distribution of the PlayeRank scores of the players in the match. The currently selected player is annotated with a red line. The more the line goes to the right of the plot, the better the player performed during the match according to his PlayeRank score.

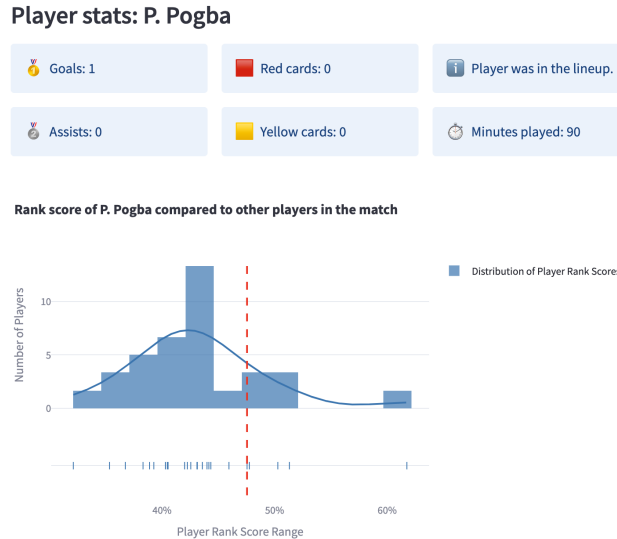


Figure 7: The statistics of the player that performed the event that was selected on the pitch.

### 3.3.4 Game Analyzer: Simulator.

The final feature of this app is the **Simulator**, to be found under the Game Analyzer as well. This feature provides the user with a play button that displays a simulation that goes through the game, minute by minute. The user can alter the play speed, time window, and total simulation time using sliders. To clarify, the time window is the window of time within which the events are made visible at each step of the simulation. If, for example, the time window is 5 minutes, then the simulation will start by showing the events that happened between minutes 0 and 5, and will increment this range to 1 and 6, 2 and 7, etc. This will continue until the total simulation time is reached. When running the simulation, a ‘Stop simulation’ button will appear to allow the user to stop the simulation beforehand if desired. Figure 8 shows an example of this.

When running a simulation, the match settings, filter settings, and color settings that were specified in the settings selection menu in the sidebar are respected.

As for the implementation, this game simulator was made possible by using the empty container feature in Streamlit. First, an empty container is created using `streamlit.empty()`. Then, this container is filled with a Plotly visualization of the game at the start of the simulation time. Finally, with every iteration, the container can be updated with a new plot.

## 3.4 Design decisions

As explained in Section 1, this application was developed as part of a project for the course *Information Visualization*, a Master’s level course given at the VUB in the Master of Engineering in Computer Science. During the lectures of this course, various theoretical concepts and design principles have been presented. In our project, we have tried to incorporate a variety of these different concepts and principles. This section will mainly focus on this aspect, explaining how these concepts and principles are applied in our project.

The first design principle applied in this project is the idea of ‘*overview first, details on demand*’. We use interactivity through hover and on-click user actions to allow for control over visualization detail and complexity. This helps in providing highly detailed and complex information without





Figure 8: The **Simulator**. In this example, the user simulates the simple passes of the entire Croatia team in red at a speed of 1, with a time window of 1 and a total simulation time of 60.

overwhelming the user, enabling a seemingly simple user experience. Additionally, our use of Plotly allows the user to zoom in on plots if desired.

We tried our best to apply design principles related to human perception and color theory. Although the default colors for arrows are initialized randomly, we provide the user with the ability to choose the colors of all arrow types in the pitch plot. This allows the user to set colors that are contrasting enough for a comfortable viewing experience. If we could, we would have manually hard-coded default colors that are spaced well and deliver a clear visual experience. However, there are so many arrow types possible that it becomes infeasible to hard-code default colors, hence the use of random colors. Additionally, we wanted to give a very thin border to the arrows to make them more visible, but this is not possible in a visually elegant way using Plotly.

Another design principle we tried to incorporate is to not use unjustified 3D or 2D. Our visualizations are at most 2D. The use of 2D is justified by the fact that we display positional data. However, for basic player characteristics such as weight and height, we provide a 1D on-hover visual. As for the 2D ranking plot, we justify this with the fact that it is the most intuitive way to compare the ranking of a player against their peers. We kept dimensionality at a minimum and only used higher dimensions where it is truly effective.

## 4 Evaluation

For the evaluation of this application, we performed a User Study with 8 participants. We have been able to perform the user study with 6 football fans, 1 coach, and 1 scout.

### 4.1 Target Audience

#### 4.1.1 Scouts

Our tool is meant to be used by scouts. The goal is to help them to find and assess players that have an interesting play style through the use of the visualization tool, with the aim of potentially recruiting them.

### 4.1.2 Sports Enthusiasts

Although regular sports fans are not the main target of our visualization tool, they can enjoy our app as a fun and interesting tool to use in a recreative manner. It will allow them to analyze the strategies and play styles of different teams. Some of these sports enthusiasts may potentially be into sports betting for example, which could make our app even more interesting for them.

### 4.1.3 Coaches

Coaches, like the scouts, should benefit a lot from this visualization since it will help them to analyze the strategies and play styles of different teams. It can be helpful to analyze which players have similar play styles or even to analyze the play styles of opposing teams.

## 4.2 Questionnaire

During the user study, the users had to perform a task. The task was to analyze the match of France against Croatia in the final of the World Cup of 2018. They had to perform several sub-tasks based on this match. After that, they also had to rate the task, by answering certain questions and by giving an approximation of the time it took to complete the task. The sub-tasks and the questions asked were the following:

### Sub tasks

- What do you think the strategy of both teams is in terms of offensiveness?
- Which player performed the best in both teams?
- Who scored during the match?
- Which team do you think had the highest successful pass average?
- Which team do you think had the highest ball possession?

### Rating (scale of 1 to 5)

- How easy do you think it is to analyze the strategy of a team during a match?
- How easy is it to find a player you would potentially recruit?
- How intuitive is the app to use?
- How often did you get confused?
- How useful do you think the app is in your life as a fan/scout/coach?
- How do you like the responsiveness of the app?

### Features

- What features would you like to add to the tool?
- Would you modify some of the features in the tool? If so, what would you change?

## 4.3 Results

The results of the evaluation show that users believe the tool was helpful to visualize and analyze a given player's strategy. It is also easy to analyze the dynamics of a whole team. The participants could easily infer based on the visualization what the strategies were of each team pretty accurately. The participants also all answered correctly that the Croatian team had the most average successful passes and that it also had the highest ball control. Overall the analysis of the team's strategies and the dynamics of the teams are done pretty rapidly. The average times to complete the tasks are the following: 1.5 minutes to complete the first sub-task, 3.4 minutes for the second one, 4 minutes for the third one, 1 minute for the fourth one, and 0.9 minutes for the last one. We can conclude from these results that analyzing team dynamics and strategies is pretty easy using the tool because it can be done quite quickly (sub-tasks 1, 4, and 5). However, finding information

about who performed the best or who scored during the match is slower. In the questionnaire, participants also had to rate the app, the results of which are as follows: the first question received an average rating of 4.6, the second one received an average rating of 3.1, the third one got an average rating of 4.6, the fourth one got an average rating of 1.5 (the lower the better), the fifth question got an average rating of 4 and the last question got an average rating of 4.8. These results show us that the app succeeded in making analyzing the strategies of different teams easy. It performs less well in terms of finding a player to recruit, but it is still a feasible task within a time that isn't unreasonable. The app was experienced as intuitive and responsive, and it was not confusing the users. Most of the participants also thought that the app could be useful in their life, but perhaps this is biased toward football fans since we did not have a lot of candidates for the other positions.

## 5 Conclusion and Future Work

The goal of this project was to build a visualization app that aids football coaches, scouts, and fans with the analysis of football matches and football players. The aim is to provide them with a way to gain insights into the play styles of certain players, and patterns in the match, allowing them to determine how to coach specific players or whether or not to recruit a particular player.

The question to ask now is whether or not we achieved our target. Based on the evaluation, we conclude that the main target of our application has indeed been reached. Both the coach and scout in our evaluation sample stated that this application provides them with valuable insights into the analysis, recruitment, and strategy development of football teams. Our application provides value to the end user in an interesting way. It is not only a tool for consuming data in an intuitive way, but it actually also provides a way to discover *new* insights into the match that go beyond statistical information. Examples are the ability to infer the aggression of a team, the pass flow success throughout a particular time window, or even the actual dynamics and strategy of the team (e.g., long-range passes immediately after ball possession to leverage the new advantage gained). We, therefore, consider our app a relative success.

However, based on the feedback we received and our own ideas, there are some points that could be addressed in future work. The following sections provide a few examples of what such future work could entail.

### 5.1 Compact Color Selection

Currently, the color selection menu in the app becomes quite large when all events are selected for visualization. This phenomenon is not aesthetically pleasing. The color selection menu becomes so large because Streamlit components are often not very flexible in how they can be displayed, and we did not manage to create a compact selection menu. Ideally, we would want to make the color picker box smaller, and have the label of each color picker placed next to the box rather than above it.

### 5.2 Player Role Filter & Name on Rank Hover

In its current state, the application is very good at analyzing a given player. However, it does not provide a fast way for a user to *find* the best player in a particular role. For instance, finding the best defender in a particular match requires selecting all players to be displayed in the **Player Stats** menu, then going through each player to check their rank score and their role.

To solve this, two things could be done. Firstly, a role filter could be added in the filter section, to allow the user to search players by role. Secondly, hovering over a particular data point in the ranking plot should not only show the value of the rank but also the name of the player. These two steps would make the search for specific players faster.

### 5.3 Player Highlighting

At the moment, it is possible to visualize the events of certain players independently, and it is possible to visualize the events of a whole team. However, when visualizing the events of a whole team, there is no way to then highlight a specific player among the team arrows. This is something that could be added to make it possible to analyze the effect a certain player has on the whole team. For example, a right-click on an event that was performed by a particular player will then color all the other player's events in a contrasting color from the rest of the team arrows.

Furthermore, it is currently not possible to display information about a certain player without clicking on an event that was performed by the player. Indeed, a user must click on an event first before being able to see the more extensive player statistic details. As this information is not really related to a particular event, it should be possible to view it, without having to first find an event performed by the player that we want to see.

### 5.4 Action Highlighting

In our visualization, there is no way of distinguishing *actions* from each other. An action is a sequence of events performed by the same team that forms a perfect chain with no interruptions by the other team. We have performed the pre-processing step to annotate the different events with a certain action number, but our visualization does not make use of this information at the moment. This is something that could be very useful because it gives us more insights into the sequential structure of events during the game.

### 5.5 Simulation Navigation

Currently, the only ways to navigate the `Simulator` are by either playing through it entirely or by stopping the simulation prematurely. Future work could include extending the ways that the simulation can be interacted with, by for instance adding a pause button, buttons to move frame by frame both forward and backward, etc.

## References

- Pappalardo, L., Cintia, P., Ferragina, P., Massucco, E., Pedreschi, D., & Giannotti, F. (2019). Playrank: Data-driven performance evaluation and player ranking in soccer via a machine learning approach. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(5), 1–27.